# **IST/RT** EZ-BIST User Manual

# v3.4.1



# Contents

1.	Introduction to EZ-BIST	1
	1.1. Features	1
	1.2. Architecture	2
2.	EZ-BIST Command Options and Parameters	3
	2.1. Invoke EZ-BIST with the GUI Mode	4
	2.2. Input Verilog Files	5
	2.3. Specify the Working Path	7
	2.4. Auto-Identify the Memory Model	8
	2.5. The Generate the ROM Signature	9
	2.6. Template File Generator	9
	2.7. Input BFL File	10
	2.8. Insert MBIST to Design	10
	2.9. Specify Top Module	10
	2.10.Disable Clock Tracing	12
	2.11. Input UDM File	12
	2.12.Generate UDM File in GUI Mode	13
	2.13.Integrate Multiple MBIST Circuits	18
	2.14.Generate UDM File with Library File	18
	2.15.Generate UDM File with Configuration File	19
	2.16. Parsing Type Definition	20
	2.17.Fault Free	20
	2.18.RCF Generator	21
	2.19.STIL Format	21
3.	EZ-BIST BFL Options	22
	3.1. OPTION Function Block	22
	3.2. BIST Function Block	36
4.	EZ-BIST Output Files	60
	4.1. Self-MBIST Related Files	60
	4.2. Insert MBIST Related Files	61
	4.3. Generate Folders	62
	4.4. Makefile	63
	4.5. Macro File	65
5.	BII File	67
	5.1. Integrator Function Block	67
	5.2. Testbench Function Block	76
6.	Appendixes	81
	6.1. "Include" Case	81

Copyright ©2023-2024 iSTART-TEK INC. All rights reserved.



6.2.	Parsing Mode	81
6.3.	*.rcf File	81
6.4.	Supported Testing Algorithm	82
6.5.	Statistics in TSMC SP Memory	85
6.6.	RTL Syntax Restrictions	90



# List of Figures

Figure 1-1	EZ-BIST Operation Flow Diagram	. 2
Figure 2-1	EZ-BIST Command Options	. 3
Figure 2-2	EZ-BIST GUI Mode	. 4
Figure 2-3	Verilog File Path	. 5
Figure 2-4	File-list File Example	. 6
Figure 2-5	Work Path	. 7
Figure 2-6	Memchecker Information	. 8
Figure 2-7	The Example of *_gold_signature.txt	. 9
Figure 2-8	EZ-BIST Template Generator	. 9
Figure 2-9	Top Module Name	.11
Figure 2-10	User Defined Memory	12
Figure 2-11	Open UDM GUI	13
Figure 2-12	Support Batches Adding and Multiple Formats	14
Figure 2-13	Memory Parameter Settings	15
Figure 2-14	IO Editing through EZ-BIST	16
Figure 2-15	IO Adding Rapidly Using Drag & Drop	16
Figure 2-16	Delete IO with Right Click	17
Figure 2-17	User Define Memory Generation	17
Figure 2-18	UDM Configuration File Example	19
Figure 3-1	OPTION Function Block	22
Figure 3-2	Block Diagram of System Design with MBIST Inserted	24
Figure 3-3	Clock Sub Function Block	27
Figure 3-4	Group Function Block	29
Figure 3-5	Open Memory Info File	31
Figure 3-6	Example of Memory Info File	32
Figure 3-7	Support Batches Adding and Multiple Formats	32
Figure 3-8	Memory Info Setting Information	33
Figure 3-9	PHYSICAL Sub Function Block	34
Figure 3-10	MBIST Function Block	36
Figure 3-11	Example of Synchronous/Asynchronous Circuit	39
Figure 3-12	Example of ATPG Circuit	40
Figure 3-13	Commands for Programmable Algorithm Function	40
Figure 3-14	The Example Loop Test Waveform	42
Figure 3-15	Example of Retention Time Option in testbech.v	46
Figure 3-16	Implementation of Bypass Circuit by Wire	48
Figure 3-17	Implementation of Bypass Circuit by Register	48
Figure 3-18	Example of Register Sharing	49



Figure 3-19	Clock Architecture of clock_function_hookup Option	50
Figure 3-20	Clock Architecture of clock_switch_of_memory Option	50
Figure 3-21	Diagnosis Fail Memory Information	51
Figure 3-22	Diagnosis Fail Time Information	51
Figure 3-23	Default Algorithm Function Block	53
Figure 3-24	select_elem_testing	
Figure 3-25	Select Testing Elements Sub Function Block	55
Figure 3-26	BFL TechNode	57
Figure 3-27	BFL Setting File	58
Figure 3-28	Run the BFL Setting File	59
Figure 4-1	Clock Gating Logic for Simulation and Synthesis	
Figure 4-2	Clock Gating Cell with Waveform	
Figure 5-1	Load BII	67
Figure 5-2	Options of Integrator Function Block	
Figure 5-3	Hookup Sub Function Block	71
Figure 5-4	BII File Hookup Information Table in *.integ File	72
Figure 5-5	The Example of Port Connection	73
Figure 5-6	The Example of Wire Connection	74
Figure 5-7	Group Sub Function Block	75
Figure 5-8	Testbench Function Block	77
Figure 5-9	Initial_sequence Sub Function Block	
Figure 5-10	Example of BII Setting Content	79
Figure 5-11	Run BII Setting File	80
Figure 5-12	The Status Window When BII Flow is Completed	80



# List of Tables

Table 1-1	EZ-BIST Features	1
Table 1-2	EZ-BIST Input Files	2
Table 1-3	EZ-BIST Output Files	2
Table 3-1	Clock Information	28
Table 3-2	Commands for Programmable Algorithm	41
Table 3-3	BG Field Definition	43
Table 3-4	Example of Bit Inverse	43
Table 3-5	Example of Column Inverse	44
Table 3-6	Example of User-defined Background and Test Pattern	45
Table 3-7	Supported Units of Retention Time	47
Table 3-8	Fixed Four Memory Addresses	52
Table 3-9	Fixed Two Memory Addresses	52
Table 3-10	Format of March CW Element	56
Table 4-1	Self-MBIST Related Files	60
Table 4-2	Insert MBIST Related Files	61
Table 4-3	Generated Folder	62
Table 4-4	Commands of Makefile	63
Table 6-1	Testing Algorithms for SRAM in EZ-BIST	82
Table 6-2	Testing Algorithms for ROM in EZ-BIST	84
Table 6-3	The Default Setting of BFL file	85
Table 6-4	Synthetic Area of default.bfl	86
Table 6-5	Area Comparison Table	88



# Type conversion in this document

Conversion	Meaning for use
Bold	Items in the user interface that you select or click and text that you type into the user interface
<italic></italic>	Variables in commands, code syntax, and path names
Courier	File name
66.99	Emphasize the meaning
Color in blue	The outputs from EZ-BIST tool presenting in blue color
	Omitted material in a line of code
÷	Omitted lines in code and report examples
[]	Optional items in syntax descriptions to specify
()	Explanations or to clarify meaning
{}	Repeatable items in syntax descriptions
	Separated the individual item in syntax descriptions



# 1. Introduction to EZ-BIST

EZ-BIST is an EDA tool that can generate the test circuit for MBIST (Memory Built-In Self-Test), providing total solutions including comprehensive test algorithms, autogrouping mechanism, and auto-integration mechanism for MBIST circuits and the original circuit. It is easy for users to generate optimized MBIST circuits.

### 1.1. Features

As shown in Table 1-1, EZ-BIST supports several features. For more details, please refer to <u>Application Notes</u>.

	Feature	Description
РОТ	Power_On-Test	It is used to guarantee that memory can execute normally after powered on, EZ-BIST supports the POT function for users to implement the POT design.
АСТ	Auto-Clock Tracing	ACT can trace the clock root to the clock source of memory modules and classify those memories into different clock domains. This mechanism not only saves time of connecting clock sources manually but also helps users to trace the clock in an easier way during creating MBIST.
BUF	Bottom-Up Flow	BUF is designed for IP/Harden implementation. Users can insert MBIST in an individual module. Then, integrate these individual modules in the top module.
AGC	Auto-Gating Clock Cell Insertion Flow	To reduce power consumption, EZ-BIST supports AGC for users to insert gate cells and MUX in front of MBIST automatically.
DIAG	Diagnosis Function	In general, MBIST only shows the results of pass or fail after MBIST executes memory testing. To analyze memory defects, EZ-BIST supports memory diagnosis to collect related information such as memory failure addresses, failure patterns, etc. In addition to collecting information, EZ-BIST diagnosis can also assign diagnosis buffer sizes and control the diagnosis timing.

Table	1_1	F7-BIST	Features
lane	1-1		i eatures



### 1.2. Architecture

Figure 1-1 shows the operation flow of EZ-BIST.



Figure 1-1 EZ-BIST Operation Flow Diagram

EZ-BIST input files include the files listed below:

Top HDL Design	Top HDL design with memory models
Memory Module	Verilog files of memory models
UDM Files	User-defined memory files

EZ-BIST output files include the files listed below:

Table 1-3 EZ-BIST Output Files		
Inserted Design	Integrated MBIST circuits with the top HDL design	
Synthesis Scripts	Synthesis scripts for users to synthesize	
MBIST Verilog Design	Generated MBIST circuits design	
Fault Memory	Generated fault memory models	
	This is used to verify functional correctness of MBIST and	
	circuits with a pre-defined error bit memory.	
Testbench	Testbench of MBIST circuits simulation	

Copyright© iSTART-TEK INC Limited 2023-2024, All Rights Reserved.



# 2. EZ-BIST Command Options and Parameters

Users can execute EZ-BIST commands with the options, *--help* or *-h*, to know all the options supported by EZ-BIST. Figure 2-1 shows an example of executing EZ-BIST with option *-h* and this chapter will introduce these options. The upper section is the command list. The lower section is the command descriptions.

```
usage: ezBist [-h] [-bii INTEGRATE FILE] [-bfl BFL FILE]
             [-f RUN_FILE [RUN_FILE ...]] [-v VERILOG_FILE [VERILOG_FILE ...]]
             [-W DIR] [-top MODULE] [-I] [--genmeminfo]
             [-integ FILE [FILE ...]] [-u FILE [FILE ...]] [-pm Verilog type]
             [--integrator] [--faultfree] [--ug UDM FILE config FILE]
             [--rcfg Addr length Data width output FILE] [--tempgen]
             [--memchecker] [--memlib2udm MEMLIB_FILE]
             [--bflconfig [BFL FILE]] [--biiconfig [BII FILE]]
             [--pathconv work path] [--STILloopformat work path]
             [--latchgo_hier latchgo_data meminfo] [--udmgui [UDMGUI]]
             [--meminfogui [MEMINFO]]
optional arguments:
-h, --help
                                        show this help message and exit
-bii INTEGRATE FILE
                                        input BII file
-bfl BFL FILE
                                        input BFL file
-f RUN FILE [RUN FILE ...]
                                       input run file(s)
-v VERILOG FILE [VERILOG FILE ...] input 4 verilog file(s)
-W DIR
                                        specify working path
-top MODULE, -T MODULE
                                        specify top module
                                        insert BIST to design
-I, --insert
                          .....
```

### Figure 2-1 EZ-BIST Command Options

Copyright© iSTART-TEK INC Limited 2023-2024, All Rights Reserved. November, 2023



### 2.1. Invoke EZ-BIST with the GUI Mode

Usage: --gui

**Description:** This option is used to invoke EZ-BIST with the GUI mode.

**Example:** \$ *ezBist* --*gui* 



Figure 2-2 EZ-BIST GUI Mode



### 2.2. Input Verilog Files

	BFL Configuration Tool _ 🗆
e Option	
TION CLOCK GROU	BIST EZ-BIST TechNode BFL content
verilog path	./run.f
user define memory	
user denne memory	
top module name	top
top hierarchy	top
🗹 auto group	
clock trace	
✓ insertion	
integrator mode	
Work path	./mbist
fault free	
parsing mode	RTL_only 3
block path	
□ force system verilo	g
memory library	
ecc prefix	
Log	
Log	[16:24:45] [START] Compiling done (1.73 sec)
	[16:24:45] [VERILOG] All messages about verilog analyzing is in "verilog_analyze.log".
	[16:24:45] [START] Total execution time : 4.01 sec
Status	
	100%

Figure 2-3 Verilog File Path

Usage: -v [VERILOG\_PATH]

**Description:** This option specifies the paths of Verilog design files. The design files here include "system design files" and "memory models". EZ-BIST provides an auto-insertion function to integrate MBIST circuits into the original system design. For this reason, users need to provide the whole design files rather than the memory files only.

This option supports either reading one Verilog file or reading all files in the working directory. It also supports the file-list file format  $* \, . \, \pm \, . \, \, \pm \, . \, \, \pm \, . \, \,$ 



- Example 1:\$ ezBist -v vlog\_1EZ-BIST will read the Verilog files in vlog\_1 directory.
- Example 2: \$ ezBist -v vlog\_1/file1.v vlog\_2/file4.v EZ-BIST will read the file1.v in vlog\_1 directory and file4.v in vlog\_2 directory.
- **Example 3:** \$ *ezBist -v filelist.f* EZ-BIST will read the designs in fielist.f. Figure 2-4 is an example of the file-list file.

-v ./memory/rf\_2p\_24x28.v
-v ./memory/sram\_sp\_4096x64.v
-v ./memory/rom\_6144\_64.v
-v ./memory/rf\_sp\_128x22.v
-v ./memory/sram\_dp\_1024x64.v
-v ./memory/rf\_2p\_24x56.v
-v ./memory/sram\_sp\_2048x64.v
-v ./memory/sram\_sp\_640x32.v
-v ./memory/rf\_2p\_64x64.v
-v ./memory/rf\_2p\_72x14.v
-v ./memory/sram\_sp\_1024x32.v
-v ./memory/RA1RW\_D2048\_W128\_BE\_RE.v
-v ./memory/RA1RW\_D1024\_W128\_BE\_RE.v
-v ./memory/RA1RW\_D1024\_W128\_BE\_RE.v
.v ./memory/RA1RW\_D1024\_W128\_BE\_RE.v
.v ./memory/RA1RW\_D1024\_W128\_BE\_RE.v
.v ./memory/RA1RW\_D1024\_W128\_BE\_RE.v

### Figure 2-4 File-list File Example



### 2.3. Specify the Working Path

Usage: -W [WORK\_PATH]

- **Description:** This option is for setting the output directory of EZ-BIST execution results.
- **Example 1:** \$ *ezBist* -*v* [*VLOG\_PATH*]/[*file\_1*].*v* -*W* [*WORK\_PATH*] EZ-BIST will read the file\_1.v design file and save output results into WORK\_PATH.
- **Example 2:** \$ *ezBist -v* [*VLOG\_PATH*]/[*file\_1*].*v* Without the -*W* option, EZ-BIST will save all generated results into the current working directory.

BFL Configuration Tool _ 🗆 ×					
File Option					
OPTION CLOCK GROUP	BIST EZ-BIST TechNode BFL content				
warile a nath					
verlig path	./run.r				
user define memory					
top module name	top				
top hierarchy	top				
🗹 auto group					
clock trace					
✓ insertion					
✓ integrator mode					
Work path	./mbist				
fault free					
parsing mode	RTL_only 🗘				
block path					
force system verilog					
memory library					
ecc prefix					
Log	[16:24:45] [START] Compiling done (1.73 sec)				
	[16:24:45] [VERILOG] All messages about verilog analyzing is in "verilog_analyze.log".				
	[16:24:45] [START] Total execution time : 4.01 sec				
Status					
	100%				

Figure 2-5 Work Path



### 2.4. Auto-Identify the Memory Model

Usage:	memchecker
Description:	This option is used to execute EZ-BIST memory checker to identify memory models defined by users with the <i>-v</i> option.
Example:	<i>\$ ezBistmemchecker -f filelist.f</i> Users can check if there is a memory model that cannot be identified by reviewing the output messages as Figure 2-6.

### Input file(s):

[1] /home /workspace/project/memchecker/memory/rom 6144 64.v [2] /home//workspace/project/memchecker/memory/rf 2p 24x56.v [3] /home//workspace/project/memchecker/memory/sram sp 4096x64.v [4] /home//workspace/project/memchecker/memory/sram sp 640x32.v [5] /home//workspace/project/memchecker/memory/sram\_sp\_2048x64.v [6] /home//workspace/project/memchecker/memory/rf 2p 72x14.v [7] /home//workspace/project/memchecker/memory/RA1RW D2048 W140... [8] /home//workspace/project/memchecker/memory/RA1RW D2048 W128... [9] /home//workspace/project/memchecker/memory/sram sp 1024x32.v [10] /home//workspace/project/memchecker/memory/rf\_sp\_128x22.v [11] /home//workspace/project/memchecker/top.v [12] /home//workspace/project/memchecker/memory/sram dp 1024x64.v [13] /home//workspace/project/memchecker/memory/RA1RW\_D1024\_W128... [14] /home//workspace/project/memchecker/memory/rf 2p 24x28.v [15] /home//workspace/project/memchecker/memory/rf 2p 64x64.v Valid file(s): [1] /home//workspace/project/memchecker/memory/rom 6144 64.v [2] /home//workspace/project/memchecker/memory/rf 2p 24x56.v [3] /home//workspace/project/memchecker/memory/sram sp 4096x64.v [4] /home//workspace/project/memchecker/memory/sram sp 640x32.v [5] /home//workspace/project/memchecker/memory/sram sp 2048x64.v [6] /home//workspace/project/memchecker/memory/rf 2p 72x14.v [7] /home//workspace/project/memchecker/memory/RA1RW D2048 W140 BE RE.v [8] /home//workspace/project/memchecker/memory/RA1RW D2048 W128 BE RE.v [9] /home//workspace/project/memchecker/memory/sram sp 1024x32.v [10] /home//workspace/project/memchecker/memory/rf sp 128x22.v [11] /home//workspace/project/memchecker/memory/sram dp 1024x64.v [12] /home//workspace/project/memchecker/memory/rf\_2p\_24x28.v [13] /home//workspace/project/memchecker/memory/rf 2p 64x64.v Unrecognized file(s): [1] /home//workspace/project/memchecker/top.v

Figure 2-6 Memchecker Information

Copyright© iSTART-TEK INC Limited 2023-2024, All Rights Reserved.

November, 2023



### 2.5. The Generate the ROM Signature

- Usage: --memchecker
- **Description:** This option is used to execute the EZ-BIST memory checker to generate a golden ROM signature with the -*v* [ROM memory RTL code file] option.
- **Example:** \$ *ezBist --memchecker -v* [*ROM memory RTL code file*] Users can verify the signature created by the MBIST and compare with the golden one.
  - \$ ezBist --memchecker -v rom\_6144\_64.v
- Note: The value of a signature will be saved in the \*\_gold\_signature.txt file (see Figure 2-7) and in the meantime, a top.v file will be generated and replaced the previous one in the memory folder.

rom\_6144\_64\_verilog gold\_signature = 7be4eb

Figure 2-7 The Example of \*\_gold\_signature.txt

### 2.6. Template File Generator

- Usage: --tempgen
- **Description**: This option is used to generate a template file of EZ-BIST. These template files include BII (MBIST Integration Information) files, BFL (MBIST Feature List) files, UDM files, and PGF files as Figure 2-8.
- **Example:** \$ *ezBist* --*tempgen*

[ezBist][TEMPLATE] ezBist template generator:

- 1. BIST Feature List (BFL)
- 2. BIST Integration Information (BII)
- 3. User defined memory
- 4. Pattern Gen File (PGF)
- 5. Quit

[ezBist][TEMPLATE] Select an option (Enter ':q' to quit):

Figure 2-8 EZ-BIST Template Generator

Copyright© iSTART-TEK INC Limited 2023-2024, All Rights Reserved.

November, 2023



### 2.7. Input BFL File

Usage: -bfl BFL\_FILE
Description: This option is used to define a BFL file for EZ-BIST.
Example: \$ ezBist -bfl [filename].bfl -W [WORK\_PATH] After executing this command, EZ-BIST will base on the parameter setting in the [filename].bfl file to generate MBIST related files into WORK\_PATH.

### 2.8. Insert MBIST to Design

Usage:	-I,insert
Description:	This option is used to integrate the generated MBIST circuits into the original system designs. Users need to define a top module name with the <i>-top</i> option when using this option.
Example:	\$ ezBist -I -top [TOP_MODULE] -v [VLOG_PATH]/[file_1].v

### 2.9. Specify Top Module

Usage: -top [TOP\_MODULE]
 Description: This option is used to integrate the generated MBIST circuits into the original system designs. Users need to define a top module name with the -top option when using this option.
 Example: \$ ezBist -I -top [TOP\_MODULE] -v [VLOG\_PATH]/[file\_1].v



	Al an			BFL Configuration	1 Tool _
	tion				
OPTION	CLOCK	GROUP	BIST	EZ-BIST TechNode	BFL content
verilog	g path		./run	.f	
user d	efine me	morv			
top me	odule nar	ne	top		
top hie	erarchy		top		
⊠ aut	o group				
	ck trace				
⊻ ins	ertion				
⊠ inte	egrator n	iode			
Work	path		./mbi	st	
🗌 fau	lt free				
parsin	g mode		RTL_0	only	•
block	path				
🗌 for	ce systen	n verilog			
memo	ry library	,			
ecc pr	efix				
Log			[16:2	4:45] [START] Com	piling done (1.73 sec)
			[16:2 "veril	4:45] [VERILOG] All og_analyze.log".	messages about verilog analyzing is in
			[16:2	4:45] [START] Total	execution time : 4.01 sec
Status					
					100%
			_		

Figure 2-9 Top Module Name



### 2.10. Disable Clock Tracing

Usage:	-N,disabletracedclk
Description:	This option is used to disable the clock tracing function of EZ-BIST. The default setting is "enabled".
Example:	\$ ezBist -N -Itop [TOP_MODULE] -f file_list.f

### 2.11. Input UDM File

Usage: -u UDM\_FILE

- **Description:** This option is used to read the UDM files generated by users. Users can generate UDM files when EZ-BIST cannot identify memory models automatically. To edit a UDM file, please refer to <u>Application Notes</u> for details.
- **Example:** \$ *ezBist -bfl [filename].bfl -u \*.udm -W [WORK\_PATH]* EZ-BIST will read BFL files and UDM files in the working directory. The output results will be saved into WORK\_PATH.

	BFL Configuration Tool _
e Option	
PTION CLOCK GROU	P BIST FZ-BIST TechNode BEL content
verling path	./run.f
user define memory	
top module name	top
top hierarchy	top
☑ auto group	
clock trace	
✓ insertion	
✓ integrator mode	
Work path	./mbist
fault free	
parsing mode	RTL_only 0
block path	
force system verilo	
memory library	-
ecc prenx	
Log	[16:24:45] [START] Compiling done (1.73 sec)
	[16:24:45] [VERILOG] All messages about verilog analyzing is in "verilog_analyze.log".
	[16:24:45] [START] Total execution time : 4.01 sec
Status	
	100%

Figure 2-10 User Defined Memory

Copyright© iSTART-TEK INC Limited 2023-2024, All Rights Reserved.



### 2.12. Generate UDM File in GUI Mode

User can choose **Open UDM GUI** directly from BFL GUI.

	BFL Config	juration	n Tool	_ 0
ile Option		-		
Close Tips				
<u>Open UDM GUI</u>		chNode	BFL content	
Open meminfo G	UI			
Open user define	d algorithm GUI	/hom	ve/istart.writer/workspace/NDAcase/ru	
101	zbist_templatei	17101.5	eristal twitter, workspace, tipreze,	<u> </u>
user define memory				
top module name				
top hierarchy				
✓ auto group				
clock trace				
✓ insertion				
integrator mode				
Work path				
□ fault free				
parsing mode	Netlist_only			0
block path				
□ force system verilog				
memory library				
ecc prefix				
Log	BFL mode: Load fil [/home/istart.wri ≥[H≥[2]	e ter/wor	rkspace/NDAcase/ezbist_template.bfl]	
	(c) Copyright 20	)09 - 20 <sup>:</sup>	021 by iSTART-Technologies, Inc.	~
Status		_		_
			100%	

Figure 2-11 Open UDM GUI



🐝 鹰用程式位置系统 😼 🍜	8 📕		🔂 👘 75	J13日(三) 16 35 istart.writ
		UDM Editor - [sram_sp_1024x32]		- *
File View Edit				00
roperty (Read-Only)	13 B		Memory Creator	6
tem	Value		CD (CD AM _ CD _ C	
write_latency	0		SP/SHAMI SRAM_2P SRAM_DP ROM	
operation_mode				
dont_touch				Update SP Param
command				
◊ io			Memory Module Name	sram_sp_1024x32
column_width	3		Address Count	1024
memory_class	SRAM_DP		Hudioss count	1024
address_count	1024		Column Width	4
▶ port				
data_width	64		Data Width	32
> rom_6144_64				
RA1RW_D2048_W140_BE_RE		— <b>P</b> A Q <b>P</b> —	Doc	pus Single-Port Ram
rf_2p_24x28		-CEN	Memory Class	SRAM/REGELE C
sram_sp_2048x64		D	moniory ordeo	
RA1RW_D1024_W128_BE_RE		- WEN	Power (mW)	0.0
123	=	-CLK		
▼ sram_sp_1024X32		- EMA	Read Latency	1 🤤
power	0.0	- RETN	Molto Latoren	
io			write Latency	
column_width	0		Port List (port a)	
address_count	1		IO List	
◊ port			10 Elat	
data_width	0		Property	U
read_latency	0			CLK
doc			Command (nop)	CEN
write_latency	0			WEN
command			60	0
memory_class	SRAM/REGFILE		WE	1 0
operation_mode				
> rf_2p_24x56			WMASK	0 0
sram_sp_4096x64				
≠ sram sp 640x32				
e de la companya de l				E
022-07-13 15:28:29,639 - INFO - [U	ser_Defined_memory] Parsed Model : RA1RW_D204	J8_W140_BE_RE		
022-07-13 15:28:29,642 - INFO - [U	ser_Defined_memory] Parsed Model : sram_sp_204	8x64		
022-07-13 15:28:29,643 - INFO - [U	ser_Defined_memory] Parsed Model : RA1RW_D102	:4_W128_BE_RE		
022-07-13 15:28:29,645 - INFO - [U:	ser_Defined_memory] Parsed Model : sram_sp_102	4x32		
2022-07-13 15:28:29,646 - INFO - [U	ser_Defined_memory] Parsed Model : rf_2p_72x14			
022-07-13 15:28:29 647 - INFO - III	ser Defined memoryl Parsed Model - rf. 2n. 24x56			
	and the second	and the second	and the second	
-				

Figure 2-12 Support Batches Adding and Multiple Formats

iST/RT

Set the parameters below through GUI:

- Memory basic parameter
- Port read/write behavior
- Test Port
- IO Port, Don't Touch Port, Repair Port

)0	Θ			
6		Memory Creator	Ø	emory Creator
		SP/SRAM SRAM_2P SRAM_DP ROM		SP/SRAM SRAM_2P SRAM_DP ROM
1	Update SP Param		Update SP Param	
	sram_sp_1024x32	Memory Module Name	sram_sp_1024x32	Memory Module Name
	1024	Address Count	1024	Address Count
	4	Column Width	4	Column Width
	32	Data Width	32	Data Width
	ous Single-Port Ram	Doc	pus Single-Port Ram	Doc
	SRAM/REGFILE   \$	Memory Class	SRAM/REGFILE   \$	Memory Class
	0.0	Power (mW)	0.0	Power (mW)
	1	Read Latency		Read Latency
	0	Write Latency	0	Write Latency
		Port List (port a)		Port List (port a)
	A 🔷	IO List	A	IO List
	read_write \$	Property	read_write	Property
		Command (nop)		Command (nop)
	1 2	CS	1	cs
	1 2	WE	1 0	WE
	0 \$	WMASK	0 0	WMASK
		Command (pseudo_r)		Command (pseudo_r)
	0 \$	CS	0 0	CS
	0 0	WE	0 0	WE

Figure 2-13 Memory Parameter Settings

(For the detailed information, please refer to Chapter 10 in Application Notes)



	UDM Editor - [sram_sp_1024x32]		_ = ×
File View Edit			
Property (Read-Only)		Memory Creator	gx
Item Value		SP/SRAM SRAM 2F	SRAM DP ROM
RAIRW_D···			10
read_I··· 1			Update SP Param
power 0.0 # mW		Momony Modulo Na	
doc ARM, Synchronous Singi		Wernory Wodule Na	=
▶ dont_t···		Address Count	0
▶ comm···		Column Width	0
マ io		Data Width	
Þ D		Data Midai	
Þ CLK		Doc	
♦ CEN N. MEN	- WEN	Memory Class	SRAM/REGFILE \$
Þ Q	-aclk	Power (mW)	0.0
colum···· 4			
memor···· SRAM/REGFILE		Read Latency	0
> port		Write Latency	0
data_··· 32		Port List (port a)	
▶ rf_2p_72x…		IO List	
v sram sp …		Proporty	road write
read_I··· 1		rioperty	read_write
power 0.0 # mW		Command (nop)	
write I··· 0		CS	0
◊ dont_t···			
▶ comm…			
Log			0 8
2022-07-13 15:13:06,467 - INFO - [User_Defined_memory] Par 2022-07-13 15:13:06,468 - INFO - [User_Defined_memory] Par	irsed Model : RAIRW_D2048_W140_BE_RE arsed Model : sram sn 2048xi44_BE_RE		Ê
2022-07-13 15:13:06,470 - INFO - [User_Defined_memory] Part	arsed Model : RA1RW_DD024_W128_BE_RE		
2022-07-13 15:13:06,472 - INFO - [User_Defined_memory] Part 2002 07 13 15:13:06,472 - INFO - [User_Defined_memory] Part 2002 07 13 15:13:06 473	irsed Model : sram.sp_1024x32		
2022-07-13 15:13:06,475 - INFO - [User_Defined_memory] Part 2022-07-13 15:13:06,475 - INFO - [User_Defined_memory] Part	aseu mudu : 1catat ased Mudu : [catat66		
2022-07-13 15:13:06,477 - INFO - [User_Defined_memory] Par	arsed Model : RA1RW_D2048_W128_BE_RE		=
0000 07 40 45 40.00 400 1050 10-0 0-44			



	UDM Editor		×
File View Edit			
Property (Read-Only)	X	Memory Creator	0 8
Item         Value           b rf_2D_24x         b           P RAIRW_D         b           b sram_sp         b           b rf_2D_72x         b           b RAIRW_D         b           b RAIRW_D         b           b RAIRW_D         b           b rd_2D_72x         b           b rd_2D_72x         b           b rd_2D_72x         b		SP/SRAM SRAM_2P SRAM_DP ROM Memory Module Name Address Count	Update SP Param
9 sram_5p p rom_514 p 122 p RALRW_D··· p rf_2p_24 p sram_5p p sram_5p		Column Width Data Width Doc Memory Class	0 0
▶ rf_2p_64x…	sram_sp_1024x32	Power (mW)	0.0
	- OA Q O- - OCEN - OD - OWEN	Read Latency Write Latency Port List (port a) IO List Property	0
	-OCIK -OEMA -ORETN	Command (nop) CS WE WMASK	
Log 2022-07-13 15:28:29.639 - INPO - [User_Defined_memory] Parss 2022-07-13 15:28:29.642 - INPO - [User_Defined_memory] Pars 2022-07-13 15:28:29.643 - INPO - [User_Defined_memory] Parss 2022-07-13 15:28:29.646 - INPO - [User_Defined_memory] Parss	d Model : RAIRW_D2048; W140_BE_RE d Model : sram_sp_2048;64 d Model : sram_sp_1024;8;64 d Model : sram_sp_1024;82 d Model : sr_2p_72:14	-	2 8

Figure 2-15 IO Adding Rapidly Using Drag & Drop



-		UDM Editor - [sram_sp_1024x32]		_ = >
File View	Edit			
Property (Read-O	bnly) 🔯	Me	emory Creator	0
Item	Value		SP/SPAM CRAM OR CRAM DR DOM	
◊ rf_2p_24x…			ST/SIGNI SRAM_2P SRAM_DP ROM	
▶ RA1RW_D…				
D sram_sp				Update SP Param
D RA1RW D			Memory Module Name	
read_I…	1		Address Count	0
power	0.0 # mW		Column Width	0
doc	ARM, Synchronous Single-Port Ram			
write_I···	0		Data Width	0
p dont_t···			Doc	
⊽ io				
► A	1		Memory Class	SRAM/REGFILE
ÞC	delete	- WEN	Power (mW)	0.0
▷ CLK		- CLK		
▷ CEN		- EMA	Read Latency	0
♦ WEN			Write Latency	0
₽ Q			Part list (and a)	
memoru	4 SPAM / PEGEILE		Port List (port a)	
addres···	2048	1	IO List	
▶ port			Property	read write
data	64		Topolo	Tead_write
▶ rf_sp_128…			Command (nop)	
			CS	
read_I···	1		00	
p embed	0.0 # mW		WE	0 0
doc	ARM, Synchronous Dual-Port Ram		WMACK	
write_I···	0		WMASA	· · ·
> operati…				
Log				0
2022-07-13 15	:28:29,639 - INFO - [User_Defined_memory] Pa	ed Model : RA1RW_D2048_W140_BE_RE		
2022-07-13 15	:28:29,642 - INFO - [User_Defined_memory] Pa	ed Model : sram_sp_2048x64		
2022-07-13 15	:28:29,643 - INFO - [User_Defined_memory] Pa	ed Model : RA1RW_D1024_W128_BE_RE		
2022-07-13 15	:28:29,645 - INFO - [User_Defined_memory] Pa	ed Model : sram_sp_1024x32		
2022-07-13 15	:20.29,040 - INFU - [User_Defined_memory] Pa :28:29 647 - INFO - [User_Defined_memory] Pa	ed Model: 11_2P_12x14 ad Model: 11_20_24x56		



	set width = 1
define{module}[sram_sp_1024x32]	end_define{io}
set address_count = 1024	
set column width = 4	define{io}[0]
set data_width = 32	set alias = Q
set doc = ARM, Synchronous Single-Port Ram	set hold time = 0 # ns
set newory_class = SRAM/REGFILE	set mux = no
set power = 0.0 # mw	set property = Q
set reau_tatency = 1	set type = output
	set width = 32
define{port}[porta]	end_define{io}
set io list = A,CEN,CLK,D,Q,WEN	
set property = read write	
end define{port}	define{dont_touch}[EMA]
	set alias = EMA
4-61(1-)(4)	set force_to = 001
derine(10)[A]	set type = input
set allas = A	set width = 3
set nota_time = 0 # ns	end_define{dont_touch}
set mux = yes	
set property = ADDR	define{dont touch}[RETN]
set type = input	set alias = RETN
end define/io}	set force_to = 1
end_deline(io)	set type = input
	set width = 1
define{io}[D]	end_define{dont_touch}
set alias = D	
set hold_time = 0 # ns	
set mux = yes	
set property = D	define (commend) (cond)
set type = input	define{command}[read]
set width = 32	set UE = 1
end_define{io}	end define{command}
define{io}[( K]	
set active = 1	define{command}[write]
set alias = CLK	set CS = 0
set hold time = $0 \#$ ns	SET WE = 0
set mux = no	end_deline{command}

Figure 2-17 User Define Memory Generation

Copyright© iSTART-TEK INC Limited 2023-2024, All Rights Reserved.



### 2.13. Integrate Multiple MBIST Circuits

Usage:	integrator
Description:	This option is used to integrate multiple MBIST circuits.
Example:	<i>\$ ezBistintegrator -bii [filename].bii -W [WORK_PATH]</i> EZ-BIST will refer to BII files to integrate multiple MBIST circuits and save output results into WORK_PATH.

### 2.14. Generate UDM File with Library File

Usage:	memlib2udm -lv [filename].memlib or _memlib2udm -f		
	[filename].memlib		
Description:	This option is used to generate UDM files from memory libra		

- **Description:** This option is used to generate UDM files from memory library files. If there is only one file, use --*memlib2udm* -*lv* [*filename*].*memlib*. If there is a file list that contains multiple files, use --*memlib2udm* -*f* [*filename*].*memlib*.
- Example:\$ ezBist --memlib2udm -lv sram\_512x8.memlibEZ-BIST will generate UDM files for memory sram\_512x8.



### 2.15. Generate UDM File with Configuration File

**Usage:** --ug UDM\_File config\_file

**Description:** This option is used to generate UDM files based on the settings in the configuration file. The configuration file is used to set different widths for address port and data port. Figure 2-18 shows an example of the configuration file. The first column defines the memory model name, the second column defines the address count, the third column defines data width, and the fourth column defines mux.

Example: \$ ezBist --ug sram\_512x8.udm config.file EZ-BIST will generate UDM files with the same type as the sram\_512x8 memory model but with different data width or address width.

#module_name	address_count	data_width	mux	
U40LP_VHD_SRF_16X8M4B1	16	8	4	
U40LP_VHD_SRF_116X38M4B1	116	38	4	
U40LP_VHD_SRF_216X28M4B1	216	28	4	
U40LP_VHD_SRF_316X18M4B1	316	18	4	

Figure 2-18 UDM Configuration File Example



### 2.16. Parsing Type Definition

Usage:	-pm,parsingmode
Description:	This option is used to specify the input design type. The supported types are <b>RTL_only</b> and <b>Netlist_only</b> .
Example:	<pre>\$ ezBist -pm Netlist_only -v example.v EZ-BIST will import example.v with the nestlist format.</pre>

### 2.17. Fault Free

### Usage: --faultfree

- **Description:** This option is used to decide whether the generated system designs include fault memory modes or not. When this option is set, the system designs with and without fault memories will be generated. When this option is not set, only the system designs with fault memories will be generated. The file name will be [design]\_INS.v.
- Example 1: \$ ezBist -bfl ezBist \_template.bfl -I -W ./work EZ-BIST will generate an integrated system design with fault memory models.
- Example 2: \$ ezBist -bfl ezBist \_template.bfl -I --faultfree -W ./work EZ-BIST will generate integrated system designs with and without fault memory models, respectively.



### 2.18. RCF Generator

Usage:	rcfg address_length data width output_file		
Description:	This option is used to generate an example RCF file for ROM memory model. The content of output RCF file is random.		
Example:	\$ ezBistrcfg 32 8 example.rcf EZ-BIST will generate an example RCF file with 32x8 matrix format.		

### 2.19. STIL Format

Usage:	STILloopformat
Description:	Change STIL file into the loop format.
Example:	\$ ezBistSTILloopformat
	EZ-BIST will generate STIL file into loop format.
	If there are many repetitive testing commands, using the option will
	simplify the testing commands as loop instructions.



# 3. EZ-BIST BFL Options

Users can execute EZ-BIST to generate the MBIST circuits with the BFL flow. This chapter will introduce the setting options in the BFL file.

The definitions of function blocks in BFL file are defined as follows: *define{function}* 

end\_define{function}

Users can find different options in each function block as below.

### **3.1. OPTION Function Block**

Figure 3-1 shows the parameters in the OPTION function block.

				BFL Confi	guration	Tool	-
ile <u>O</u> p	tion						
OPTION	CLOCK	GROUP	BIST	EZ-BIST Te	chNode	BFL content	
verilog	path						
user d	efine me	mory					
top mo	odule nar	ne					
top hie	erarchy						
🗆 aut	o group						
🗆 clo	ck trace						
🗆 inse	ertion						
🗆 inte	grator n	node					
Work	path						
🗆 fau	lt free						
parsin	g mode		Netlist	t_only			
block p	path						
🗆 fore	ce systen	n verilog					
memo	ry library	1					
ecc pro	efix						
Log							
Charles							
Status							

### Figure 3-1 OPTION Function Block

Argument	Option			
Description				
verilog_path	User defined			
Set the Verilog file paths for EZ-BIST. Th <i>fileN.v</i> or file-list file (* . f).	e format can be set either by <i>file1.v</i>   <i>file2.v</i>			
Note: Each file is separated by a vertical b	par " ".			
Example: set verilog_path = ./top.f				
user_define_memory	User defined			
Set UDM file paths for EZ-BIST. The format can be <i>memory1.udm</i>   <i>memory2.udm</i>     <i>memoryN.udm</i> . Note: Each file is separated by a vertical bar "]".				
For more details, please refer to <u>Applicatio</u>	<u>on notes</u> .			
Example: set user_define_memory = BRAINS.udm				
top_module_name	User defined			
Set the top module name of the system de	esign which includes memory modules.			
Example: set top module name = top				
top_hierarchy	User defined			
Specify the location (instance name) of the controller for MBIST circuits in the design architecture. Example: set top_hierarchy = top				
clock_trace	No, Yes			
This option is for users to disable/enable the clock source tracing function. The default setting is "no".				
No:Disable the clock source tracing functionYes:Enable the clock source tracing function				

Copyright© iSTART-TEK INC Limited 2023-2024, All Rights Reserved.



Argument	Option		
Description			
auto_group	No, Yes		
This option is for users to automatically g the GROUP function block. The default se	roup memory models based on the settings in tting is "no".		
No: Disable the clock auto-grouping to Yes: Enable the clock auto-grouping f	function unction		
insertion	No, Yes		
This option is used to integrate the gene designs. Figure 3-2 shows the block diagra	erated MBIST circuits and the original system am of the inserted system design.		
No:Disable the insertion functionYes:Enable the insertion function			
Top Design with connections to BIST MCK MEN RST IS IS IS IS IS Input signal for top design IS OUTPUT Signal for top design IS IS IS IS IS IS IS IS IS IS			
Figure 3-2 Block Diagram of System Design with MBIST Inserted			
integrator_mode	No, Yes		
This option is for users to add the dedicated testing port in the top module of MBIST. Because these testing ports adhere to standard protocols such as IEEE 1149.1, users can use the shared pin design to reduce the pin count. The default setting is "no".			
No: EZ-BIST will generate some spect them to control MBIST or get dat	cific hookup pins for the BII flow. Users can use a from MBIST.		
Yes: EZ-BIST will reserve signals into flow.	ernally in advance for testing only in the BFL		
Note: The option must be set to "yes" when clock tracing turns on.			
work_path	User defined		
Specify the path for saving the generated results in the BFL flow.			

### Copyright© iSTART-TEK INC Limited 2023-2024, All Rights Reserved.

November, 2023



Argument	Option			
Description				
fault_free	No, Yes			
When this option is set to "no", EZ-BIST fault memory models. On the contrary, we generate two integrated system designs simulation will run on without fault memory system design.	will generate an integrated system design with when this option is set to "yes", EZ-BIST will with and without fault memory. However, the MBIST circuits are integrated into the original			
parsing_mode	RTL_only, Netlist_only			
This option defines the file format of th Netlist_only.	e imported design, supporting RTL_only and			
Note: If the Netlist file are not uniquified, the	ne parsing mode must be set to "RLT_only."			
ecc_prefix	User defined			
Specify the prefix of ECC (Error Correction	n Code) related files.			
For example, when this option is set to "EC like ECC_[design]_INS.v and ECC_[f	CC", the output repair-related files will be named ilename]_tb.v etc.			
memory_library	User defined			
Define memory library (shown in Example 1) to make START to load the information of memory models. If multiple files need to be set, they can be separated by a vertical bar (' '). Alternatively, users can also fill in the memory file list as shown in Example 2.				
Example 2: set memory_library = ./mem_lib.f				
block_path	User defined			
While the design is implemented with the bottom-up flow to insert MBIST into the sub module, it will generate a *.blockinfo file in the sub module. Example: set block nath = /block1/START block1 blockinfo   /block2/START block2 blockinfo				
force system verilog	No. Yes			
The parsing format will be changed to System Verilog when users set the option to "yes". The default setting is "no".				
No: Initial parsing format is Verilog. Yes: Changed parsing format to System Verilog.				

Copyright© iSTART-TEK INC Limited 2023-2024, All Rights Reserved.



Argument	Option	
Des	cription	
disable_auto_identify	No, Yes	
The default setting is "no".		
When the user confirms that the "set memory_library" for the memories in the design has been input in the tool, enabling "disable_auto_identify" will deactivate the tool's "auto identify memory" feature to reduce the overall runtime of the tool.		
skip_check_translate_off	No, Yes	
Under the default condition (skip_check_translate_off = no), START will skip analyzing the content between "//synopsys translate_off" and "// synopsys translate_on":		
//synopsys translate_off (content) //synopsys translate_on		
If users want the tool to recognize skip_check_translate_off to yes.	e and analyze the content, please set	



### 3.1.1. CLOCK Sub Function Block

Users can define the information of clock domain or provide an SDC file for EZ-BIST to do clock tracing.

ile Option  PPTION CLOCK GROUP BIST EZ-BIST TechNode BFL content  SDC file  Clock Domain  domain name clock_domain_1  clock cycle (ns) 100.0  clock source list  Insert Delete Prev Next  define{Clock}  define{ =      define{clock_domain_1}  set clock_cycle = 100.0     set clock_domain_1}  define{clock_domain_2}  set clock_source_list =     end_define{clock_domain_2}  set clock_source_list =     end_define{clock_domain_2} end_define{clock_domain_2} end_define{clocK}					BFL Co	onfigurati	on Tool				_ 0
OPTION CLOCK GROUP BIST EZ-BIST TechNode BFL content   SDC file   Image: Clock domain Image: Clock domain Image: Clock domain Image: Clock domain   Insert Delete Prev Next   define{CLOCK}  set clock_domain set clock_domain set clock_domain set clock_domain set clock_domain set clock_gource list = end_define{Clock_domain end_define{Clock_domain set clock_source list = end_define{Clock_domain end_define{ClocK} set clock_source list = end_define{ClocK} domain Next	ile <u>O</u> p	tion									
SDC file Clock Domain domain name clock_domain_1 clock cycle (ns) 100.0 clock source list Insert Delete Prev Next define{CLOCK} set sdc_file = end_define{clock_domain_1} set clock_source_list = end_define{clock_domain_1} define{clock_domain_2} set clock_source_list = end_define{cLock_domain_2} end_define{CLOCK}	OPTION	CLOCK	GROUP	BIST	EZ-BIST	TechNod	e BFL	. content			
SDC file Clock Domain domain name clock_domain_1 clock cycle (ns) 100.0 clock source list Insert Delete Prev Next define{CLOCK} set sdc_file = define{clock_domain_1} set clock_source_list = end_define{clock_domain_1} define{clock_domain_2} set clock_source_list = end_define{cLOCK} end_define{CLOCK}		1									
Clock Domain domain name clock_domain_1 clock cycle (ns) 100.0 clock source list Insert Delete Prev Next define{CLOCK} set sdc_file = define{clock_domain_1} set clock_source_list = end_define{clock_domain_1} define{clock_domain_2} set clock_cycle = 100.0 set clock_source_list = end_define{clock_domain_2} set clock_source_list = end_define{clocK} domain_2} end_define{clocK}	SDC fi	le									
domain name       clock_domain_1         clock cycle (ns)       100.0         clock source list	Clock Do	main									
clock cycle (ns) 100.0 clock source list Insert Delete Prev Next define{CLOCK} set sdc_file = define{clock_domain_1} set clock_cycle = 100.0 set clock_source_list = end_define{clock_domain_1} define{clock_domain_2} set clock_cycle = 100.0 set clock_source_list = end_define{clock_domain_2} end_define{clocK}	domai	n name	clock_d	domain	_1						
clock source list Insert Delete Prev Next define{CLOCK} set sdc_file = define{clock_domain_1} set clock_source_list = end_define{clock_domain_1} define{clock_domain_1} define{clock_domain_2} set clock_source_list = end_define{clock_domain_2} end_define{CLOCK}	clock	ycle (ns	) 100.0								
Insert Delete Prev Next define{CLOCK} set sdc_file = define{clock_domain_1} set clock_cycle = 100.0 set clock_source_list = end_define{clock_domain_1} define{clock_domain_2} set clock_cycle = 100.0 set clock_cycle = 100.0 set clock_domain_2} end_define{clock_domain_2} end_define{CLOCK}	clock s	source lis	st 🗌								
Insert     Delete     Prev     Next       define{CLOCK}     =          set sdc_file     =          set clock_cycle     = 100.0          set clock_source_list     =          end_define{clock_domain_1}           define{clock_domain_2}           set clock_source_list     =           end_define{clock_domain_2}           end_define{clock_domain_2}           end_define{clock_domain_2}           end_define{clock_domain_2}											
<pre>define{CLOCK} set sdc_file =     define{clock_domain_1}     set clock_cycle = 100.0     set clock_source_list =     end_define{clock_domain_1}     define{clock_domain_2}     set clock_source_list =     end_define{clock_domain_2} end_define{clock_domain_2} </pre>		Insert			Delete			Prev		Next	
<pre>define{CLOCK} set sdc_file =     define{clock_domain_1}     set clock_cycle = 100.0     set clock_source_list =     end_define{clock_domain_1}     define{clock_domain_2}     set clock_source_list =     end_define{clock_domain_2} end_define{clocK_domain_2}</pre>									 		
	defin set end_ defin set end_ end_def	e{clock_c clock_s clock_s define{cl e{clock_c clock_c clock_s define{cl ine{CLO	domain_1 ycle : ock_doma domain_2 ycle : surce_list ock_doma CK}	} = 100.0 = ain_1} } = 100.0 = ain_2}	,						

Figure 3-3 Clock Sub Function Block

Argument	Option				
Descr	iption				
sdc_file	User defined				
Specify the path of an SDC file.					
define{clock_name}	User defined				
Set the clock domain name.					
clock_cycle	User defined				
Set the operating period of clock domain defined in "clock_name".					
clock_source_list	User defined				
Set the source pin or port of clock domain defined in "clock_name".					

### Table 3-1 Clock Information



### **3.1.2. GROUP Sub Function Block**

EZ-BIST assigns memory grouping according to the rule of clock domains, types of memory models, the criteria of grouping specifications, and power consumption. Users can also do memory grouping manually based on their own project requirements by editing the memory information file \*.meminfo. Memory models in the same group can be tested in parallel to reduce the testing time.

Each memory will have the dedicated SEQ\_ID (Sequencer ID) and GRP\_ID (Group ID). Memories have the same SEQ\_ID and GRP\_ID are in the same group and can be tested at the same time.

The SEQ\_ID is classified by types, specifications, and the clock domains of memory models. This ID means which sequencer the memory models belong to. The GRP\_ID is classified by power consumption and number limitations of a single group.

BIST E2-BIST TechNode BFL content     GROUP   sequencer limit 60   group limit 30   30 *   memory list   time hierarchy 0.5   lib path   power limit (mW) 1.0   hierarchy limit 0     PHYSICAL   enable physical   physical location file   distance limit   1			BFL Configuration	Tool	
OPTION CLOCK GROUP BIST EZ-BIST TechNode BFL content   GROUP   sequencer limit   50   group limit   30   memory list   time hierarchy   0.5   lib path   power limit (mW)   1.0   hierarchy limit   0	ile Option				
GROUP  sequencer limit 60  group limit 30  memory list time hierarchy 0.5  lib path power limit (mW) 1.0 hierarchy limit 0  PHYSICAL enable physical physical location file group limit 1  physical logical 0.5	OPTION CLOCK GR	OUP BIST	EZ-BIST TechNode	BFL content	
sequencer limit 60 group limit 30 memory list time hierarchy 0.5 lib path power limit (mW) 1.0 hierarchy limit 0 PHYSICAL enable physical distance limit 1 physical logical 0.5	GROUP				
group limit       30         memory list          time hierarchy       0.5         lib path          power limit (mW)       1.0         hierarchy limit       0         PHYSICAL          enable physical          distance limit       1         physical logical       0.5	sequencer limit	60			
memory list ime hierarchy ib path power limit (mW) i.0 hierarchy limit 0  PHYSICAL enable physical physical location file imit imemory limit i	group limit	30			
time hierarchy 0.5 lib path power limit (mW) 1.0 hierarchy limit 0 PHYSICAL enable physical physical location file distance limit 1 physical logical 0.5	memory list				
lib path power limit (mW) 1.0 hierarchy limit 0 PHYSICAL enable physical physical location file distance limit 1	time biorarchy	0 E			
power limit (mW) 1.0 hierarchy limit 0 PHYSICAL enable physical physical location file distance limit 1 physical logical 0.5	time merarchy	0.5			
power limit (mW) 1.0 hierarchy limit 0	lib path				
hierarchy limit 0	power limit (mW)	1.0			
PHYSICAL physical location file distance limit physical logical 0.5	hierarchy limit	0			
physical location file	PHYSICAL				
distance limit 1	physical location file	e			
physical logical 0.5	distance limit	1			
	physical logical	0.5			

Figure 3-4 Group Function Block

Copyright© iSTART-TEK INC Limited 2023-2024, All Rights Reserved.


Argument		Option			
	Description				
sequencer_limit		User defined			
This option defines the max	kimum amount o	of memory instances in a sequencer.			
Default Value: 60					
group_limit		User defined			
This option is used to defir number should be less than	ne the maximum In the value of <b>se</b>	amount of memory instances in a group. This equencer_limit.			
Default Value: 30					
memory_list		User defined			
Specify the paths of memo info file.	ry info file (* .m	eminfo). Figure 3-8 is an example of memory			
For more details, please re	fer to <u>Applicatio</u>	n Notes.			
lib_path		User defined			
This option is for users to information of memory mod based on the power criteria	o set the path dels from *.lil through the <b>po</b>	of memory libraries. EZ-BIST will load power of files and do memory grouping automatically wer_limit option.			
time_hierarchy		0 (time) <= value <= 1 (hierarchy)			
This option is for users to a The default value is 0.5.	This option is for users to adjust the weight between the testing time and design hierarchy. The default value is 0.5.				
For example:					
set time_hierarchy = 0	<i>set time_hierarchy = 0</i> EZ-BIST will assign memory grouping based on the optimized testing time. The testing time will be the highest priority.				
set time_hierarchy = 1	assign memory grouping by hierarchy ne logical hierarchy will be the highest priority.				
power_limit		User defined			
Set the maximum limitation	of power consu	Imption in one group.			
For example: <i>set power_limit = 0.005</i>					
Note: The unit is mW and c	an be decimal.				

Copyright© iSTART-TEK INC Limited 2023-2024, All Rights Reserved.

November, 2023



Argument	Option		
Description			
hierarchy_limit	User defined		
Set the maximum hierarchy number when doing auto-grouping. If the hierarchy number between memory models is larger than this number, EZ-BIST will not group these memor models into the same group.			
Default Value: 0 (no limitation of hierarchy number)			

As shown in Figure 3-5, users can open a memory info file by clicking the "File" menu and selecting "Open".





Figure 3-6 is an example of the memory info file. For the detailed information, please refer to Chapter 7 in <u>Application Notes</u>.

M iSTART-MemInfo	_ = ×
File Edit	
	A
- DOMANI Internetion default	- mining 100 0ncl
<ul> <li>Loomaini <ul> <li>Loomaini <li>Loomaini <li>Loomaini <li>Loomaini </li> </li></li></li></ul> </li> </ul>	, cycle=100.01isj
▼ [SE0] [no=1,memory clas	ss=SRAM/REGFILE1
	· · · · · · · · · · · · · · · · · · ·
[SP=2_2_2, byp=no, diag=no, q_pipe=no, repair=no] sram_sp_1024x32 top.u_t1.ram_1	
[SP=2_2_2, byp=no, diag=no, q_pipe=no, repair=no] sram_sp_1024x32 top.u_t1.ram_2	
[SP=2_2_2, byp=no, diag=no, q_pipe=no, repair=no] sram_sp_1024x32 top.u_t1.ram_3	
[SP=2_2_2, byp=no, diag=no, q_pipe=no, repair=no] sram_sp_1024x32 top.u_t1.ram_4	
[SP=2_2_2, byp=no, diag=no, q_pipe=no, repair=no] sram_sp_1024x32 top.u_t1.ram_e	
[SP=2_2_2, byp=no, diag=no, q_pipe=no, repair=no] sram_sp_1024x32 top.u_t1.ram_w	<mark>۵</mark> د د د د د د د د د د د د د د د د د د د
[SP=2_2_2, byp=no, diag=no, q_pipe=no, repair=no] sram_sp_1024x32 top.u_t1.ram_x	
[SP=2_2_2, byp=no, diag=no, q_pipe=no, repair=no] sram_sp_1024x32 top.u_t1.ram_y	
[SP=2_2_2, byp=no, diag=no, q_pipe=no, repair=no] RAIRW_D1024_W128_BE_RE top.u_t1.u_r1_ram	
[SP=2_2_2, byp=no, diag=no, q_pipe=no, repair=no] RAIRW_D20248_W128_BE_RE top.u_t1.u_r2_ram	
[SP=2_2_2, oyp=no, diag=no, d_pipe=no, repair=noj kalkw_D2048_w140_BE_KE top.u_t1.u_r3_ram	
• [SEQ] [n0=2,memory_ctax	ss=sram_2Pj
• [unour] [10-2]	
(21 - 2_5_3, 0)p-10, 0(g=10, 0_p)p=10, (epail=10) 11_2p_24x20 (0,0-3)	SC-SRAM DP1
← [GR0][P] [n=3]	33-5104h_01]
[DP=2.4.4, byo=no, dia#=no, o pipe=no, repair=noi sram do 1024x64. top.u 11.u do u	-
IDP=2.4.4 hvm=nn diad=nn o nine=nn renair=nni sram dn 1024x64 ton u tit u dn2	
###Total mbist memory instance = 14	
##Total SRAM/REGILE = 11	
###Total SRAM 2P = 1	
###Total SRAM_DP = 2	
###Total REPAIR SRAM/REGFILE = 0	Ξ.
###Total REPAIR SRAM_2P = 0	
###Total REPAIR SRAM_DP = 0	
###Total ROM = 0	
###top_cerautr algoritom is= (March C+ @2P;SOULD)(March C+,SOULD)(March C+ @DP;SOUD)	
**************************************	

Figure 3-6 Example of Memory Info File

As shown in Figure 3-7, users can right click "GROUP" and select "add mem" to add memories by batches according to the information described below.

BB/XXAABQ			
KEY	VALUE	MemoryEdit	×
- [DOMAIN]	[name=top_default, cycle=100.0ns]	Memory Type	SP 0
· [CTR]		By Pass	yos • no
- [SEO]	[no=1.memory class=SRAM/REGFILE]	Diagnosis	• yes 👘 no
IGROUPI	[no=1]	Q Pipe	🔶 yes 🔅 no
[SP=2, 2, 2, byn=no, diag=no, q, pipe=no, repair=no] sram, sp. 1024x32	top u t1 ram 1	Repair Mode	yes • no
$[SP-2, 2, 2]$ by p-no, diag-no, q_pipe-no, repair-no] stam_op_1024x32	top u t1 ram 2	Memory Module Name	
$[SP=2,2,2,0]$ by p=no, diag=no, q_pipe=no, repair=no] statt_sp_1024x32	top.u_t1 ram 2	Memory instances	_
[3F-2_2_2, byp-no, diag-no, q_pipe-no, repair-no] statt_sp_t0z4x32	top.u_t1.ram_5		
[SP=2_2_2, byp=no, diag=no, q_pipe=no, repair=noj sram_sp_1024x32	top.u_t1.ram_4		
[SP=2_2_2, byp=no, diag=no, q_pipe=no, repair=no] sram_sp_1024x32	top.u_t1.ram_e		
[SP=2_2_2, byp=no, diag=no, q_pipe=no, repair=no] sram_sp_1024x32	top.u_t1.ram_w		
[SP=2_2_2, byp=no, diag=no, q_pipe=no, repair=no] sram_sp_1024x32	top.u_t1.ram_x		
[SP=2_2_2, byp=no, diag=no, q_pipe=no, repair=no] sram_sp_1024x32	top.u_t1.ram_y		
###Total mbist memory instance = 14			
###Total SRAM_2P = 1			
###Total SRAM_DP = 2 ###Total REPAIR SRAM/REGFILE = 0			
###Total REPAIR SRAM_2P = 0 ###Total REPAIR SRAM_DP = 0			
###Total ROM = 0			_
######################################		<u>Cancel</u> <u>OK</u>	
2022-07-15 15:46:35,365 - INFO - open .meminfo file >> /home/istart.writer/workspace/NDAcase/mbist/E2-BIST_memory_spec.meminfo 2022.07.15 15:51:32 377 - W4RNING - open .meminfo file fall >>			(A)
2022 07-15 15:54:52,089 - INFO - Clear Table			=
2022-01-12 12:24:36,864 - INFO - Open Imenitio IIIe >> /none/istar.witer/workspace/muk/case/mole//22-0131_menion/_spec.menimo			

Figure 3-7 Support Batches Adding and Multiple Formats

A memory info file includes the following items. For the detailed information, please refer to Chapter 7 in <u>Application Notes</u>.

- Clock domain: It shows "memory clock domain name" and "testing clock cycle".
- Memory module: It shows the "memory module name" and "memory hierarchy".
- **Bypass**: Set the values of the bypass function.
- **Diagnosis**: Set the values of the diagnosis function.
- **Q\_pipeline**: Set the value of the Q\_pipeline option.
- **Group Architecture**: This option shows the grouping architecture information including the controller, sequencer, and group.
- **Design information**: This option shows the number of memory instances, memory types, and testing algorithms.



Figure 3-8 Memory Info Setting Information



# 3.1.3. PHYSICAL Sub Function Block

File Ontion	BFL Configuration Tool _
OPTION CLOCK GR	OUP BIST EZ-BIST TechNode BFL content
GROUP	
sequencer limit	60 O
group limit	30
memory list	
time hierarchy	0.5
lib path	
power limit (mW)	
biorarchy limit	
merarchy mint	v
DHYSTCAL	
enable physical	
physical location fi	
diatanaa limit	•
pnysical logical	0.5

Figure 3-9 PHYSICAL Sub Function Block

(For the detailed information, please refer to the table in the next page.)



Argument		Option		
Description				
enable_physical		No, Yes		
If this option is set to "yes", Information) file.	EZ-BIST will aut	to-group based on the DEF (Design Physical		
physical_location_file		User defined		
Set the paths of the DEF file	e.			
controller_scope		User defined		
After editing a SCOPE file, be included with a controlle	set the path of th r name and posit	ne SCOPE file. The scope information should ion coordinate as follows.		
Controller Name Position C	oordinate (x1 y1)	) (x2 y2)		
For example, top_default (1	For example, top_default (10000 10000) (300000 400000)			
physical_logical 0 <= value <= 1		0 <= value <= 1		
This option is to adjust the weight between physical coordinates and values defined in the <b>time_hierarchy</b> option.				
For example:				
set physical_logical = 0	EZ-BIST will calculate the number of intermediates based on an internal algorithm. Memory models which are located near this intermediate number will be merged into the same group.			
<i>set physical_logical</i> = 1 EZ-BIST will execute memory grouping based on the value of the <b>time_hierarchy</b> option.				



## **3.2. BIST Function Block**

TION	CLOCK	GROUP	BIST	EZ-BIST TechNo	ode	BFL content	
ption	user_de	fine_algo	rithm				
□ s	TIL test b	pench				diagnosis support	
🗆 w	GL test b	pench				diagnosis data sh	aring
🗹 as	synchron	ous reset				diagnosis memory	/ info
bist i	nterface		ieee1	500 🗘		diagnosis time inf	o
🗆 ad	dd addre:	ss y			dia	agnosis faulty item	- -
algoi	rogram a rithm sel	lgorithm ection	no		un	<ul> <li>✓ algorithm</li> <li>✓ operation</li> <li>✓ element</li> <li>✓ sea id</li> </ul>	<ul> <li>✓ grp_id</li> <li>✓ address</li> <li>✓ ram_data</li> <li>✓ rom_data</li> </ul>
back	ground s	tyle	S	OLID 🔅			
🗆 ba	ackgroun	d bit inve	rse				
bypa	ss suppo	ort emory disa	able	n ()		parallel on reduce address si rom result shiftou Q pipeline	mulation It
bypa	ss reg sh	haring	G	٢		Power-On Testing Hardwire ROM Dvnamic Memorv	Testina
🗆 cl	ock func	tion hook	up				
⊻ cl □ cl □ cl	ock swite ock sour ock with	ch of men ce switch in pll	nory			ELC FUNCTION	

Figure 3-10 MBIST Function Block

Argument	Option				
Descript	Description				
STIL_test_bench	No, Yes				
Generate a test pattern with the STIL format (IEEE 1450-Standard Test Interface Language) for the tester machine when this option set to "yes". Since the result in the default STIL format might be a lot of repeated codes, users can change it into the loop format by using command lines, <i>STILloopformat</i> .					
No: Not generate the test pattern with the Yes: Generate the test pattern with the S	e STIL format TIL format				



Argument		Option			
Description					
WGL_test_be	WGL_test_bench No, Yes				
Generate a test this option is se	st pattern with the WGL formated to "yes".	t (Waveform Generation Language) when			
No: Not g Yes: Gene	enerate the test pattern with the erate the test pattern with the W	e WGL format /GL format			
bist_interface		basic, basicIO, ieee1500, ieee1149.1			
Select the MBI Note: For more Note: When us interface. Note: When us interface.	Select the MBIST interface. Note: For more details of these interfaces, please refer to <u>IO Pin Definition</u> . Note: When users set <b>bist_interface</b> to "ieee1149.1", then IEEE 1149.7 will be the output interface. Note: When users set <b>bist_interface</b> to "ieee1500", then IEEE 1149.1 will be the output interface.				
add_address_	_y	No, Yes			
testbench supp and Y stands f No: The N Yes: The N	testbench supports the X and Y addressing modes (X stands for the row of the memory, and Y stands for the column of the memory.) No: The MBIST pattern testing only supports the X direction. Yes: The MBIST pattern testing supports both X and Y directions.				
X_Y = 00	Write MBIST pattern in the X d	lirection only.			
X_Y = 01	Write MBIST pattern the X dire	ection first, and then Y direction.			
X_Y = 10	Write MBIST pattern in the Y d	irection first, and then X direction.			
X_Y = 11	Write MBIST pattern in the Y d	irection only.			
Note: This option does not support memories with a column width of "0". Note: To define the X or Y directions, users must modify the X_Y setting in the testbench file.					
clock_source	clock_source_switch No, Yes				
This option is used to select the testing frequency while the <b>clock_within_pll</b> option and <b>clock_switch_of_memory</b> option is turned on. The MBIST circuit will have a dedicated test input signal named TRANS. Users can use this input signal to choose the testing frequency (from SCK or MCK). Note: The option must be set to "no" when clock tracing is turned on.					

# Copyright© iSTART-TEK INC Limited 2023-2024, All Rights Reserved.

November, 2023



Argument	Option				
Description					
clock_within_pll	No, Yes				
If this option is set to "yes", the MBIST circuit will have another clock input source, SCK. This signal is used to connect with an ATE (Automatic Test Equipment) machine.					
Note: The option must be set to "no" when clo	ock tracing is turned on.				
diagnosis_support	No, Yes				
This option is used to enable the diagnosis mo time and failed memory information.	de, which can provide users with the failure				
No: Disable the Diagnosis mode Yes: Enable the Diagnosis mode					
diagnosis_data_sharing	No, Yes				
Users can integrate diagnosis circuits into the to reduce the area of MBIST circuits when this	sequencer to do diagnosis storage sharing s option set to "yes".				
diagnosis_faulty_items algorithm, operation, element, seq_id, grp_id, address, ram_data, rom_data					
This option is used to select the output items failure analysis requirement.	of the diagnosis result based on the chip				
Example: set diagnosis_faulty_items = algorithm, operation, element, seq_id, grp_id, address, ram data. rom data					
rom_result_shiftin	No, Yes				
This option is used to do ROM memory testing and import the signatures for internal verification. The scenario is used when the contents of the ROM memory is not confirmed at the initial development stage.					
For example, when users set <b>rom_result_shiftin</b> to "yes" and the <b>POT</b> function is enabled, the testing results of ROM memory will be transferred to the internal circuit via commands.					
rom_result_shiftout	No, Yes				
This option is used to do ROM memory testing and export the signatures for external verification. The scenario is used when the contents of ROM memory is not confirmed at the initial development stage.					
For example, when user set <b>rom_result_shiftout</b> to "yes" and the testing results of the ROM memory will be transferred to the output interfaces via commands.					



Argument	Option			
Description				
Q_pipeline	No, Yes			
This option is used to add an extra pipeline re	gister to MBIST logics.			
No: No extra register will be added to the	e data output of a memory model.			
Yes: An extra register will be added to the operating timing of MBIST logics.	e data output of a memory model to enhanc			
asynchronous_reset	No, Yes			
<ul> <li>The option is used to specify asynchronous or synchronous reset of MBIST. The circuit can be differentiated into two types, "synchronous reset" and "asynchronous reset". "Synchronous reset" indicates all DFFs are triggered to reset and then reset at the same time. "Asynchronous reset" indicates the reset of the circuit is based on the sequential order. In other word, this is not synchronous reset.</li> <li>No: Synchronous reset will be applied with two DFFs. In addition, hookup the RSTN port (the MBIST reset signals) and the ATPGen port.</li> <li>Yes: It indicates the asynchronous reset while one reset signal asserts.</li> </ul>				
Figure 3-11 shows an example of synchronous port is under the "scan mode", the synchronou as the asynchronous circuit to select signals.	s/asynchronous circuits. When the ATPGen is circuit will be bypassed and be regarded $\qquad \qquad $			



Argument	Option					
Description						
atpg_reset No, Yes						
This option is for users to reset the "Automat is set to "yes", EZ-BIST tool will string all th ATPG_rstn.	This option is for users to reset the "Automatic Test Pattern Generation". When the option is set to "yes", EZ-BIST tool will string all the reset signals under MBIST into a series of ATPG_rstn.					
Note: In the BII flow, hookup ATPG_rstn and Note: When users set <b>atpg_reset</b> to "yes' multiplexer (MUX) for the selection of ATP Figure 3-12.	Note: In the BII flow, hookup ATPG_rstn and ATPGen ports at the same time. Note: When users set <b>atpg_reset</b> to "yes", the ATPG signal will be inserted into the multiplexer (MUX) for the selection of ATPG_rstn or async_rstn_in signal as shown in Figure 3-12.					
$\begin{array}{c} \text{ATPG\_rstn} & \text{async\_rstn\_mux} \\ \text{async\_rstn\_in} & \text{o} & \text{o}$						
select_elem_testing No, Yes						
This option is for users to do testing with user-defined test algorithms rather than EZ-BIST built-in algorithms by controlling input interfaces. When this function is turned on, users can select the algorithm elements in the SEQ, and the elements can be tested in the testbench. A programmable algorithm is presented as a PROG entry. Figure 3-13 shows the testing commands while this option is turned on. Table 3-2 is the definition of these entries.						
PROG SEQ_ID GRP_ID	MEB_ID	BG	ALG_CMD			
<pre>SDI_Command</pre> Figure 3-13 Commands for Programmable Algorithm Function						



Table 3-2 Commands for Programmable Algorithm		
Command	Description	
PROG	PROG = 0, executing the EZ-BIST built-in algorithm PROG = 1, executing the user-defined algorithm	
SEQ_ID	Sequencer ID of the memory	
GRP_ID	Group ID of the memory	
MEB_ID	Memory ID of the memory	
BG	"SOLID" is the default background style. Only when "5A" is chosen, users can select four different modes to test. For more details, please refer to Table 3-3.	
ALG_CMD	This ALG_CMD entry is based on March algorithm, users also can define it. While PROG = 1, MBIST circuits will execute user-defined algorithms. The width of the ALG_CMD entry is based on the March element definition.	



Argument		Option		
	Description			
algorithm_	selection	No, Outside, Scan		
This option is for users to choose a single test algorithm or multiple test algorithms to test sequentially.				
No:	Users can select algorithms w sequentially.	hich will be tested with MBIST circuits		
Outside:	Users can select the test algorithm will be added when the basic inte	n with the input port ALG and this input port rface is defined.		
Scan:	Users can launch a test with IEEE	E 1149.1 or IEEE 1500.		
algorithm_	_loop_test	No, Yes		
This option is for users to improve the loop mode testing efficiency. Some tests require a loop mode, but using multiple testing commands can cause delays between the commands.				
No: Not support continuous memory testing Yes: Support continuous memory testing				
Users can send commands to control the BURN_IN signal to define the period of testing as Figure 3-14 when this option set to "yes".				
BURN_IN				
	Figure 3-14 The Example Loop Test Waveform			



Argument			O	ption
	cription			
background_style		SOLID, 5A		
The type of <b>background_style</b> can be set to "SOLID" and "5A" (Check Board), the contents are defined in the bg_table file. There is an entry named BG (Background) in the SDI_Command. When <b>background_style</b> is set to 5A, the BG settings are shown as Table 3-3.				
Note: If users adopt "Ma "5A".	arch Mdsn1" as an a	algorithm, <b>backg</b>	round	<b>d_style</b> cannot be set to
	Table 3-3 BG	Field Definition		1
	BG [1:0]		۱ ۸	
	01			
	10	5A		
	11	SOLID + 5A		
background_bit_inver	se	No, Yes		<u> </u>
Bit inverse means that the BG testing data will be inversed by the increasing order or decreasing order of the memory address. For example, the BG testing data of a 64x8 memory with SOLID BG is shown as Table 3-4. Table 3-4 Example of Bit Inverse				the increasing order or ) BG is shown as Table
Memory Address	SOLID B	G Test Data		Description
0000_0000	0000	0000_0000		g data non-inversed
0000_0001	111 <sup>-</sup>	1111_1111		g data inversed
0000_0010	000_0010 000		testin	g data non-inversed
0000_0011 1111		1_1111	testin	g data inversed



Argument			Option	
	Description			
background_col_inverse		No, Yes		
Column inverse means that the BG testing data will be inversed by the changes of the row memory address. If this changing time is larger than the CIC (Column Inverse Counts) number, the BG testing data will be inversed. The CIC number is defined by the memory Mux value.				
is shown as Table 3-5.	with Mux - +		- OOLID. THE DO LESSING VALA	
Tat	ole 3-5 Examp	le of Column Inve	erse	
Memory Address	SOLID E	G Test Data	Description	
0000_0000	000	0_000	testing data non-inversed	
0000_0001	000	0_000		
0000_0010	000	0_000		
0000_0011	000	0_000		
0000_0100	111	1_1111	testing data inversed	
0000_0101	111	1_1111		
0000_0110	111	1_1111		
0000_0111	111	1_1111		
0000_1000	000	0_000	testing data non-inversed	
0000_1001	000	0_000		
0000_1010	000	0_0000		
0000_1011	000	0_0000		



Argument			Option		
	Description				
user_define_bg		User defined			
Users can specify the backgro	ound test patte	rn via the settir	ng of <b>user_define_bg</b> .		
For example, if the width of th	e data is 4 bits	:			
Example 1: When users "SOLID", th	When users assign <b>user_define_bg</b> to "3" and <b>background_style</b> to "SOLID", then the testing pattern will be 0x3.				
Example 2: When users "5A", then t	When users assign <b>user_define_bg</b> to "3" and <b>background_style</b> to "5A", then the testing pattern will be 0x3,0xC,0x5,0xA.				
Table 3-6 lists the example of user-defined background and the corresponding test patterns.					
Table 3-6 Example of User-defined Background and Test Pattern					
Background Style Use Background Style		lefined round	Test Pattern		
SOLID		3	3		
5A	3	<u>3</u> 7	3, C, 5, A 3, C, 7, 8		



Argument		Option		
Description				
retention Handshake, Time				
This option	is for users to set the mode of	of retention.		
Handshak	Handshake: The retention time can be set in <b>retention_time</b> option in BFL file or in testbech, v file as shown in Figure 3-15.			L file or in
Time:	The retention time is fixed after being set in the <b>retention_time</b> option in the BFL file.			<b>e</b> option in
	`timescale 1ns / 1ps			
	module stimulus:			
	parameter top default boyo	2	= 100.0;	
	parameter RP default bcvc		= 100.0;	
	parameter tcyc		= 100.0;	
	parameter rcyc		= 100.0;	
	parameter cyc		= tcyc;	
	parameter CORE_ID		= {3{1'b1}};	
	parameter RP_default_RET	_time	= 5.0;	
	parameter top_default_RET	_time	= 5.0;	
	parameter TAP_IR_width		= 1*3;	
	parameter test_result_width		= 10;	
	parameter test_command_v	vidth	= 17;	
	parameter max_config_widt	h	= 17;	
	parameter WIR_width		= 6;	
	parameter COMMAND_DR_	_ID	= {2'b1, 2'b1, 2'b1};	
	parameter TEST_RESULT_	DR_ID	= {2'd2, 2'd2, 2'd2};	
	parameter max_config_widt	h	= 367;	
	parameter DIAG_RESULT_	טא_וט יייי	$- \{ \angle 03, \angle 03, \angle 03 \};$	
	parameter top_default_ALG	_width	$- \angle$ ,	
	parameter top_default_SEQ		- ∠, = 1:	
	parameter top_default_GRP		- 1,	
Figure 3-15 Example of Retention Time Option in testbech.v				



Argument				Option	
	Description				
retention_time		Use	er defined		
This option is used to define the retention time. The supported unit of retention time are listed in Table 3-7.			ed unit of retention time are		
Table 3-7	′ Supporte	d Uni	its of Retention	Time	
	Symbo	ol	Unit		
	Т		10 <sup>12</sup>		
	G		10 <sup>9</sup>		
	М		10 <sup>6</sup>		
	K or k	(	10 <sup>3</sup>		
	m		10 <sup>-3</sup>		
	u		10 <sup>-6</sup>		
	n		10 <sup>-9</sup>		
	р		10 <sup>-12</sup>		
Some memory testing algorithms allow users to do retention testing. For example, March-RET algorithm is <(wb) (SLP) <(rb) >(wa) (SLP) >(ra). The (SLP) element indicates the sleeping time is 1ms. If users want to extend the sleeping time more than 1ms, they can specify the retention time through <b>retention_time</b> .					
define{BIST}	define{BIST}				
set retention_tim	 set retention_time = 1m				
end_define{BIST}					
Note: The syntax of the rete	The syntax of the retention time has different formats.				
Take 1ms as an evan	nle				

Take 1ms as an example, The timing setting format in Verilog is **retention\_time** = 1000000 (n). The timing setting format in System Verilog is **retention\_time** = 1000000 (n) or **retention\_time** = 1m.







A	rgument	Option		
		Description		
bypass_mem	nory_disable	No, Yes		
This option is available only when <b>bypass_support</b> is enabled. The memory CS (cl select) will be disabled. For example, when CS is active high, the parameter of CS will "0". When CS is active low, the parameter of CS will be "1". All the memory clocks will tied together with "0".				
No:	The memory CS will be	enabled.		
Yes:	The memory CS will be	disabled.		
bypass_reg_	sharing	1 <= value <= 1024		
Users can set <b>bypass_supr</b> this option to	Users can set this option to define the register sharing number of bypass registers when <b>bypass_suppor</b> is set to "reg". The range is between "1 ~1024". EZ-BIST will base on this option to implement register sharing to reduce the area of bypass registers.			
For example, bits, the numb	when users assign <b>bypa</b> per of bypass registers wi	<b>ass_reg_sharing</b> to "4" and data output Q to "32" ill be "8" as Figure 3-18.		
		_XOR[0] Q[7:0]		
A D CEN WEN		_XOR[1] Q[15:8]		
		Q[31:24]		
	Figure 3-18 Ex	cample of Register Sharing		
bypass_cloc	:k	No, Yes		
If users decide to implement the bypass circuit by "reg" method, they can turn on this option to add a dedicated input port BCK for the bypass register and users can define the frequency of BCK based on their project requirements.				



Argument	Option		
De	escription		
clock_function_hookup	No, Yes		
This option is for users to hookup MCK with a memory functional clock. When this option is set to "yes", MCK will be driven by the memory functional clock directly.			
Note: The option is available only when c clock architecture of this option.	clock tracing is turned on. Figure 3-19 shows the		
Function CLK	er MCK TPG CLK er CLK Memory er of clock_function_hookup Option		
clock_switch_of_memory	No, Yes		
When this option is set to "yes", the clock signal of the memory model will be changed to MCK by clock multiplexer in the test mode. The clock signal of the memory model is running at the same frequency according to users' requirements. Figure 3-20 shows the clock architecture of this option. The MCK also can be driven by the internal testing clock. Users can hookup it with internal clock signal in BII mode.			
	Inserted Design		
MCK Function CLK			
Figure 3-20 Clock Architecture of clock_switch_of_memory Option			



Argument	Option			
Description				
diagnosis_memory_info No, Yes				
EZ-BIST will generate MBIST circuits with N-bits width LATCH_GO output signals when this option is turned on. N means the number of memory models and each bit of a LATCH_GO signal indicates one memory model. Figure 3-21 shows the waveforms of LATCH_GO signals. When the signal turns from high to low, it indicates that memory has failed.				
мск				
LATCH_GO[3:0]	<u>4'b0101</u> <u>4'b01</u> 00			
LATCH_GO[3] Error				
MBIST_top.w)				
LATCH_GO[1]	Error			
LATCH_GO[0]	Frror			
(corresponding mem_2_1_1 in MBIST_top.v)				
Figure 3-21 Diagnosis	Fail Memory Information			
diagnosis_time_info	No, Yes			
EZ-BIST will generate MBIST circuits with the MBIST_GO output signal when this option is turned on. If the memory fails, this signal will change from high to low and return to high in the next clock cycle as shown in Figure 3-22.				
Time 3,400,000 ps				
мск				
	2			
GRP_ID 1	2			
BIST_GO	r			
Figure 3-22 Diagnosis Fail Time Information				



Argument		Option			
	Description				
parallel_on		No, Yes			
Specify the testbench p	Specify the memory to support parallel testing. When this option is set to "yes" and assign testbench parameter PRL_ON to "1", all memories under a controller will launch the testing simultaneously.				
reduce_ad	dress_simulation	No, Yes			
EZ-BIST ex speeds up the testing	xecutes testing with fixed four simulation by reducing memory address will be fixed to two me	memory addresses as Table 3-8. This optic / testing addresses. If the column width is zero mory addresses as Table 3-9.	on o,		
	Table 3-8 Fixed Fe	our Memory Addresses			
	Memory Address Row	Memory Address Column			
	000000	000000			
	000000	111111			
	111111	000000			
	111111	111111			
	Table 3-9 Fixed Two Memory Addresses				
		222222			
		.111111			
not		No basic by rom rom			
Pot INO, Dasic, IIW_IOII, IOII					
more details, please refer to Chapter 9 in <u>Application Notes</u> .					
No:	Disable the POT function.				
Basic:	Basic: It indicates supporting some generic signals to enable or disable MBIST and the test results. This function only supports the RAM test				
hw_rom:	_rom: It indicates that the POT testing commands will be designed to hardwired circuits. This function supports the ROM test.				
Rom:	It indicates that the POT testing commands will be stored in the ROM. This function supports the ROM test.				



# 3.2.1. Default Algorithm Sub Function Block

EZ-BIST provides various testing algorithms for users to choose according to different testing requirements. Figure 3-23 shows the default setting of single-port memories is the March C+ algorithm. If users want to add more testing algorithms into MBIST circuits, they just need to add algorithms into this function block.

The ROM setting is used to set the algorithm for ROM, and there are two options: ROM test and ROM Test 3n.

Section 6.4 shows the testing algorithms provided by EZ-BIST.

```
define{algorithm}
    set single_port = March C+  # March C-, March LR...
    set two_port = March C+ @2P  # March C- @2P...
    set dual_port = March C+ @DP  # March C- @DP...
    set ROM = ROM Test # choose only one between ROM Test and ROM Test 3n
end_define{algorithm}
```

Figure 3-23 Default Algorithm Function Block



# 3.2.2. Programmable Algorithm Sub Function Block

As shown in Figure 3-24, users can set the programmable algorithm in the GUI mode.

BFL Confi	guration Tool _ 🗆 ×
<u>File Option</u>	
OPTION   CLOCK   GROUP BIST   EZ-BIST TechNo option   user_define_algorithm	de   BFL content
□       STIL test bench         □       WGL test bench         □       asynchronous reset         bist interface       icce1500         □       add address y         □       select elem testing         algorithm selection       no         ▼       background style         SOLID       ▼         □       background bit inverse	<ul> <li>☐ diagnosis support</li> <li>☐ diagnosis data sharing</li> <li>☐ diagnosis data sharing</li> <li>☐ diagnosis time info</li> <li>☐ diagnosis faulty items</li> <li>☐ genzilen □</li> <li>☐ genzilen □</li> </ul>
bypass support     Image: model inverse       bypass support     Image: model inverse       bypass memory disable     Image: model inverse	<pre>&gt; parallel on &gt;</pre>
bypass reg sharing 1 +	Power-On Testing no
☐ clock source switch ☐ clock within pll	

Figure 3-24 select\_elem\_testing

Figure 3-25 shows the select testing elements sub function block, describing the testing elements created by users.

-	BFL Configuration Tool _ 🗆 🗙							
File	Ор	tion						
OPT	ION	CLOCK	GROUP	BIST	EZ-BIST TechNode	BFL content		
App	plicati	on						
	Wireless_WiFi_BLE:55nm 40nm 28nm       Image: Constant of the second secon							
Sin	gle Po	ort						
	Single Port       March CW (part 1)       March CW (part 2)       March 33N       March C+       March C							
Tw	o Port	s						
	March Weak WL @2P         March CW (part 1) @2P         March CW (part 2) @2P         March 33N @2P         ✓							
Dua	al Port	ts						
	<ul> <li>March X @DP</li> <li>March Y @DP</li> <li>March LR @DP</li> <li>March 17N @DP</li> <li>March 19N @DP</li> <li>✓</li> </ul>							
alg s	algorithm programmable _support elements							
	ra, wa	a, rawb,	rawbrb ,	raraw	brb			
•	4ax pr 1	og eleme	ents Ç					

Figure 3-25 Select Testing Elements Sub Function Block



While users chose the programmable algorithm function, the ALG\_CMD entry will be added for programming usage. Users can define elements of their own testing algorithm.

For example, the March CW algorithm provided by EZ-BIST. The contents of this algorithm is  $>(wa) >(ra, wb) >(rb, wa, ra) <(ra, wb, rb) <(rb, wa) <(ra), the number of March elements is 6 and the supported elements are r, w, rw and rwr. In this case, the width of the ALG_CMD entry is 7 × 5 = 35 (5 indicates element width / EOT, End of Test should be 0) and the format definition of March element can be Direction, Parity, and Operation as Table 3-10. Users also can find the definition in the march_command.alias file.$ 

ALG\_CMD = {ALG\_CMD6, ALG\_CMD5, ..., ALG\_CMD1, ALG\_CMD0}

Туре	Field	Width	Value	Description
	>		0	Address increase
Direction	<	1	1	Address decrease
Data	а		0	Data background
Background	b	1	1	Inverse data background
	r	3	001	Read
	rw		010	Read, Write
Operation	rwr		011	Read, Write, Read
	W		100	Write

#### Table 3-10 Format of March CW Element



#### 3.2.3. BFL TechNode

To avoid the possibility of dynamic defects in electronic devices which are manufactured from the advanced processes below 50nm, more accurate algorithms are needed for memory testing. EZ-BIST provides another way to select the algorithms. According to the needs of different processes and applications, EZ-BIST TechNode will check the recommended algorithms for users as Figure 3-26.

				BFL Configuration	Tool _ □ >		
File O	otion						
	r	r					
OPTION	CLOCK	GROUP	BIST	EZ-BIST TechNode	BFL content		
Applicat	ion						
Wirel ALU:I TCON Audio Finge	ess_WiFi Programn :55nm 4 :55nm 4 rprint_re	_BLE:55n ning type Onm 28ni Onm cognition	m 40n n :180nn	m 28nm n 130nm 110nm 55	▲  =   ▼		
Single P	ort						
□ Ma □ Ma □ Ma □ Ma	arch CW ( arch CW ( arch 33N arch C+ arch C	part 1) part 2)					
Two Por	ts						
□ March Weak WL @2P       □         □ March CW (part 1) @2P       □         □ March CW (part 2) @2P       □         □ March 33N @2P       ☑         ☑ March C+ @2P       ✓							
Dual Po	rts						
March X @DP     Image: A constraint of the second sec							
algorith	m progra	mmable					
suppo	rt elemer	nts					
ra, w	a , rawb	, rawbrb ,	raraw	brb			
Max p	rog elem	ents					

Figure 3-26 BFL TechNode



# 3.2.4. BFL Setting File

Users can check the settings of the BFL file in the BFL content page.

BFL Configuration Tool _ 🗆 🗸									
File Option									
OPTION	CLOCK	GROUP	RICT	E7-BIST TechNode	BEL content	1			
OFIION	CLUCK	GROOP	<b>D131</b>	LZ-BIST TECHNOLE	Bi E content				
define{0	PTION}								
set ve	rilog_path	=		# /rela	ative path/desig	]n.f			
set us	er_define	memory	=	#	/relative path/	memory.udm			
set to	p_module_	name	=	#	design top				
set to	p_hierarch	iy =		# des	ign_top sub_m	odule_instance_name			
set clo	ock_trace	= no	)	# yes	, no (User grou	p instances will all be un-group v	v I		
set au	to_group	= n	0	# ye	s, no				
set ins	sertion	= no		# yes,	no				
set int	.egrator_n	10de =	no	# ) # /w/	ves, no		=		
set fai	ilt free	= no		# ./ WU # Ves	no.				
set na	rsing mor	1e =	Netlist	only #	RTI only Net	list only			
set blo	ock path	=	Nothot_	# /rela	tive path/block	blockinfo I			
set for	rce system	n veriloa :	= no	#	ves, no				
set m	emory_libr	rary J	=	# /r	élatíve path/me	emory.lvlib			
set ec	c_prefix =			# prefi	x for ECC relate	ed files			
define									
set	sdc file	=	-	# /relati	ve nath/design	sdc			
def	ine{clock	domain 1	}	" / Cluci	re parily design.				
	set clock	cvcle	= 100.	0 # flc	# floating point				
	set clock_s	, source_list	=	# top	# top design1 CLK				
enc	l_define{c	lock_doma	ain_1}		-				
def	ine{clock_	_domain_2	}						
	set clock_o	cycle	= 100.	0 # flo	ating point				
	set clock_s	source_list	=	# top	design2 CLK				
enc	l_define{c	lock_doma	ain_2}						
end_d	efine{CLC	CK}							
define	{GROUP}								
set	sequence	r_limit = 6	50	# in	teger				
set	group_lim	nit = 30		# inte	eger smaller tha	an sequencer limit			
set	memory_	list =		# /rel	# /relative path/design.meminfo				
set	time_hier	archy = 0	).5	<i>#</i> 0	# 0(time) < value <1(hierarchy)				
set	lib_path	=	~	# /relat	ive path/lib (Ac	cept file dictionary)			
set	power_lin	IIII = 1.1	0	# mV	# mw (float bigger than 0)				
set	nierarchy	$_{\text{infit}} = 0$		# Inte	eger (derauit: 0	)			
def	ine{PHYSI	CAL}							
9	set enable	_physical	=	no #y	es, no				
	set nhvsica	al location	file =	# /r	elative_nath/deg	sian def			
	111					(			

Figure 3-27 BFL Setting File



As shown in Figure 3-28, users can click "Run" from the "File" drop-down menu to complete the MBIST execution.

			BFL Configu	ration	Tool		_ = ×
File Option							
Load BFL							
Save BFL	GROUP	BIST	EZ-BIST Tech	Node	BFL content		
<u>R</u> un							
<u>E</u> xit ۲						 	
user define m	emory						
top module na	me						
top hierarchy							
🗆 auto group							
clock trace							
insertion							
integrator	mode						
Work path							
fault free							
parsing mode		Netlist	_only				0
block path							
force syste	m verilog						
memory librar	у						
ecc prefix							
Log							
Status							

## Figure 3-28 Run the BFL Setting File



# 4. EZ-BIST Output Files

This chapter introduce EZ-BIST's output files and their usages. These output files are divided into Self-MBIST and Inserted-MBIST. Users can use these generated files to verify the MBIST circuit, and also verify the MBIST circuit integrated with customers' own logic design.

## 4.1. Self-MBIST Related Files

The generated self-MBIST related files include the self MBIST circuits (.v), test bench (.v), file-list file (.f), synthesis script (.tcl) and brief introduction file (.html). When users run simulations with these output files, it only simulates between MBIST circuits and memories.

	EZ-BIST output	Description
Project file (.bid)	[filename]_spec.bid	This is an EZ-BIST project file and includes all settings of EZ-BIST.
Self MBIST circuits (.v)	[filename]_top.v [filename].v	<pre>[filename]_top.v includes memory models, fault memory models and MBIST circuits. It is integrated with MBIST circuits and memory models of original system design. [filename].v is HDL file of MBIST circuits.</pre>
Test bench (.v)	[filename]_tb.v	This is a test bench for testing [filename]_top.v.
File list (.f)	[filename].f	File-list file records [filename]_top.v, [filename].v and memory models. This file is used for simulation.
Synthesis script (.tcl)	[filename].tcl	This is a script file for synthesis of MBIST circuits.

## Table 4-1 Self-MBIST Related Files

## 4.2. Insert MBIST Related Files

EZ-BIST can insert MBIST circuits into customers' design. Users can verify the inserted-MBIST with their own system circuit. The following table shows the related files of the insert MBIST circuits.

	EZ-BIST output	Description
Inserted MBIST circuits (.v)	[design]_INS.v [design]_INS_f.v	The file [design]_INS.v integrate MBIST circuits with user's system designs. The [design] is the name of user's system designs. This file does not include fault memory models.
		Different from [design]_INS.v, [design]_INS_f.v is integrated with fault memory models.
Test bench (.v)	[filename]_tb_INS.v	This is a test bench for testing [design]_INS.v.
File list (.f)	[filename]_INS.f [filename]_INS_FAULT.f	<pre>File-list [filename]_INS.f records [design]_INS.v, memory models and MIBST circuits. Different from file-list [filename]_INS.f, [filename]_INS_FAULT.f also includes fault memory models.</pre>

Table 4-2 Insert MBIST Relate	ed Files
-------------------------------	----------



#### 4.3. Generate Folders

The following table shows the generated folders when executing EZ-BIST.

	EZ-BIST Output	Description
REPORT		This folder is used to save the results of synthesis.
FAULT_MEMORY	[mem_name]_f.v fault_memory.f	<pre>[mem_name]_f.v is fault memory models. Some values inside of memory are tied to 0 or 1. This is used to verify functional correctness of MBIST circuits.</pre> The file-list fault memory f records all
		generated fault memory models.

#### Table 4-3 Generated Folder

## 4.4. Makefile

EZ-BIST also generates Makefile which includes related commands of simulation and synthesis for users to verify their designs. Using Makefile, it can easily run various simulations along with MBIST circuits. Table 4-4 shows the commands of Makefile.

	Command	Description
Self-MBIST simulation	make [bistname] FUNC=tb	It is used to run self MBIST simulation with [bistname]_tb.v and [bistname].f. The simulation results will be printed out in the command line window.
Self-MBIST simulation with fault memories	make [bistname] FUNC=tb_f	It is used to run self MBIST simulation with <pre>[bistname]_tb.v, [filename].f and fault memory models. This simulation will show "Failed" because MBIST has detected faults in the memory models.</pre>
MBIST circuits synthesis	make [bistname] FUNC=dc	It is used to run synthesis with [bistname].tcl scripts using Design Compiler. The output will be saved into the REPORT folder.
Check syntax of self MBIST circuits with nLint	make [bistname] FUNC=lint	It is used to run syntax check with [bistname].f by using nLint. The checking result will be saved to file [bistname]_lint.log.
Remove generated files	make clean	It is used to remove generated files including *.log, *.fsdb,*.db, *.sdf and *.rpt files in the REPORT folder.
Inserted- MBIST simulation	make [bistname] FUNC=tb_INS	It is used to run the Inserted MBIST simulation with [bistname]_tb_INS.v and [bistname]_INS_FAULT.f, the simulation results will be printed out in the command line window. This command is available while the BFL option insertion is "yes".
Inserted- MBIST simulation with fault memories	make [bistname] FUNC=tb_INS_f	It is used to run the Inserted MBIST simulation with [bistname]_tb_INS.v, [bistname]_INS_FAULT.f and fault memory models. The simulation results will show "Failed" because MBIST has detected faults in the memory

#### Table 4-4 Commands of Makefile

Copyright© iSTART-TEK INC Limited 2023-2024, All Rights Reserved.

November, 2023



		models.
Check syntax of inserted- MBIST circuits with nLint	make [bistname] FUNC=lint_INS	It is used to run syntax check with [bistname]_INS_FAULT.f by using nLint. This checking result will be saved to file [bistname]_lint_INS.log.
Formal checking	make [bistname] FUNC=fm	It is used to run formal checking with [bistname]_fm.tcl. The output message will be saved into [bistname]_fm.log.



#### 4.5. Macro File

iSTART's latch-based clock gating cell model is  $*\_GCK.v$  (\* will be generated according to the module name in customers' designs). It can be synthesized in RTL modeling. However, to control clock skews, it is preferrable to integrate clock cells from the standard library.

**Note:** Please change each module in the macro file into the corresponding standard cell. Figure 4-1 is the example of a clock gating module. Here "ctr\_name" means the prefix name coming from the controller name in the customer's design.

```
module ctr name gck (clk out, clk en, clk in, test en);
input clk_in;
input clk en;
input test en;
output clk out;
`ifdef SYNTHESIS
    GCK_VENDOR_CELL gck(
               Q(clk_out)
               E(clk en)
               TE(test en)
               CK(clk_in)
               );
`else
        reg latch_out;
         assign clk_out = clk_in & latch_out;
         always @(clk in or clk en or test en) begin
         if (~clk_in) begin
              latch out = clk en | test en;
```
end end `endif endmodule



Figure 4-2 shows the schematic diagram of a clock gating cell with the waveform.







# 5. Bll File

EZ-BIST provides a BII (Integration Information) File for the integration task, which is in charge of integrating different MBIST controllers with an integrator module and then use IEEE1149.1 interface to communicate with ATE. This is used to save the pin count of the chip level. We will introduce the options of a BII file in this chapter.

## 5.1. Integrator Function Block

Users can define the hookup pin mapping settings and order of different MBIST controller in the following function block.

define{Integrator}[Name] ... end\_define{Integrator}

The parameter, [Name] can be modified by users, and this will be the module name of the generated integrator module. This integrator module will integrate the WSI signal and WSO signal of each MBIST controller.

Figure 5-1 shows an example to load the existing BII file as the default setting.

X BII Configuration Tool@VENU	s		×
File Option			
Load BII	h   BII Content		
Run			-1
Exit <u>Exit</u> path	INTEG		
work_path			
bist_integ_path		 	
group_order	BISTGRPO	 _	
top_module_name		_	
TAP_hierarchy		_	
nvm_ctr	×		
Group			
connection_type	broadcast		
bist_order			
Log			
Status			

Figure 5-1 Load BII



The options of the integrator function block are shown in Figure 5-2.

ntion   Alian   H	actual		
option   Allas   H	pokup		
integrator name	INTEG		
verilog_path	./work/RP_default_INS_FAULT.f	-	
work_path	./integ	-	-
bist_integ_path	./work/RP_default_spec.integ	-	-
group_order	BISTGRPO		
top_module_name	top		_
TAP_hierarchy	ltop		
nvm_ctr			
connection type	broadcast		_
bist_order	RP_default		-
Log			7

Figure 5-2 Options of Integrator Function Block



The following is the list of BII parameters and their functionalities:

Argument	Option			
Description				
group_order	User defined			
This option is for users to define the ordering of an MBIST controller by setting the group sub function block. The testing sequence will follow the setting of <b>group_order</b> .				
top_module_name	User defined			
This option is for users to define the top level module of their design.				
TAP_hierarchy	User defined			
This option is for users to define the hierarchy	y of integrator module.			
verilog_path	User defined			
For example, If the BFL option, <b>fault_free</b> is set to "yes", the generated filelist file is *_INS.f. If the BFL option, <b>fault_free</b> is set to "no", the generated filelist file is *_INS_FAULT.f.				
work_path	User defined			
Specify the path of the working directory of the BII flow. All generated files in the BII flow will be saved to <b>work_path</b> .				
bist_integ_path	User defined			
Set the path of the integration specification file *_spec.integ. Users can assign more than one integration specification files and separate them by the vertical bar " ". For example, bist_1_spec.integ   bist_2_spec.integ   bist_3 spec.integ.				
skip_include_check	No, Yes			
No: Transform all included paths in the output files into absolute paths Yes: Only transform the included paths in the modified files (which are named with keyword "_INS") into the absolute path				



Argument	Option		
Description			
serial _order	User defined		
The option is used to specify the memory testing order under the individual controller group. If the option <b>parallel_on</b> in the BFL file is "yes", the memory will be tested by one controller sequentially one after another. For some particular cases, users want to test memories under more than one controller at the same time. By using the <b>serial_order</b> option, users can assign the controller group priority testing order, and the controller group contains one or more controllers.			
For example, when users assign <b>serial_order</b> to "top_default0, top_default1   RP_default0   RP_default1" and set " <b>parallel_on</b> " to "yes". In this case: The priority testing order is [top_default0 & top_default1] => [RP_default0] => [RP_default1]			
Note: Each memory controller under a group separated by comma "," is tested at the same priority order. An individual testing controller group is separated by a vertical bar " ".			



#### 5.1.1. Hookup Sub Function Block

EZ-BIST can support to implement the hookup function automatically. When the MBIST has been completed, users can get the \*.integ file in the MBIST folder. The \*.integ file provides the hookup pins shown in Figure 5-4. Furthermore, users can define the hookup pin information and pin the remapping information in hookup sub function block.

The definitions of hookup sub function blocks in the BII file are defined as follows:

#### define{hookup}[signal]

end\_define{hookup}

hookup_name TD dedicate_port itd mapping_port top	I i ipm otdi			
Replace	Delete	Prev	Next	
define{hookup}[TDI] set dedicate_port set mapping_port end_define{hookup}	= itdi = top u_pm otdi			
define{hookup}[TMS] set dedicate_port set mapping_port end_define{hookup}	= itms = top u_pm otms			
define{hookup}[TDO] set dedicate_port set mapping_port end_define{hookup}	= itdo = top u_pm otdo			
define{hookup}[top_def	ault_MCK]			
set dedicate_port set mapping_port end_define{hookup}	= MCK = top MCK			
define{hookup}[TCK] set dedicate_port set mapping_port end_define{hookup}	= itck = top u_pm otck			
define{hookup}[TRST] set dedicate_port set mapping_port	= itrst = top u_pm otrst			

Figure 5-3 Hookup Sub Function Block



Consequently, the BII hookup information table in \*.integ file might differ depending on the user's interface.

In Figure 5-4, it shows the IEEE 1149.1 JTAG interface. EZ-BIST supports several interfaces, such as basic, basicIO, IEEE1149.7, and IEEE1149.1.

# BII flie I	hookup informatio	on table
#	interface TCK	=> define{hookup}[TCK]
#	interface TRST	=> define{hookup}[TRST]
#	interface TMS	=> define{hookup}[TMS]
#	interface TDI	=> define{hookup}[TDI]
#	interface TDO	=> define{hookup}[TDO]
#	controller clock	=> define{hookup}[top_default_MCK]
#	BIST reset in =>	define{hookup}[RSTN]
end_define	e{BIST}	
# end_define	BIST reset in => e{BIST}	define{hookup}[RSTN]

Figure 5-4 BII File Hookup Information Table in \*.integ File



Argument	Option			
Description				
hookup	User defined			
It indicates a hookup sub function block.				
signal	User defined			
It indicates the signals on the integrator module and will be connected with the mapping port. This signal could be IEEE 1149.1, IEEE 1149.7 signals, IEEE 1687 signals, MCK, or TCK. For example, the signal name in IEEE 1149.1 could be TCK, TDI, TMS, TRST, and TDO.				
dedicate_port	User defined			
Set the pin name on the boundary port of a 1149.7 signals, IEEE 1687 signals, MCK, or	chip. This port could be IEEE 1149.1, IEEE TCK.			
mapping_port	User defined			
1149.7 signals, IEEE 1687 signals, MCK, or TCK.         mapping_port       User defined         mapping_port is users' reserved port for MBIST and it can be connected to MBIST by the mode of replacing the port. The hierarchy must be specified and can be separated by a space bar. Figure 5-5 is the example of port connection.         The following is the example of command setting:         define{hookup}[TCK]         set dedicate_port = itck         set mapping_port = top u_pm otck         end_define{hookup}         top         u_pm         otck				
Eigure 5 5 The Exemple of Port Connection				



Argument	Option		
Description			
mapping_wire User defined			
It connects to MBIST through the wire assignment. The hierarchy must be specified and can be separated by a space bar. Figure 5-6 is the example of wire connection.			
The following is the example of command se	etting:		
define{hookup}[TCK] set dedicate_port = itck set mapping_wire = top u_pm o end_define{hookup}	tck		
Note: Either mapping_port or mapping_wire can be chosen.			
top			
u_pm	MBIST/MBISR		
	Assign INTEG_TCK = otck		
Figure 5-6 The Examp	le of Wire Connection		

#### **5.1.2. Group Sub Function Block**

The Group sub function block defines the grouping mechanism of all MBIST controllers.

The following syntax defines the Group sub function block.

define{group}[group\_name]
 set connection\_type = ...
 set bist\_order = ...
end\_define{group}

Note: [group\_name] should be the name which is listed in the column of group\_order.

X	BII Configuration Tool@VENU	s O	-		×
File	b Option	eb   BTT Content			
	Option   Alias   Ho	skup			1
	integrator name	INTEG			
	verilog_path	./work/RP_default_INS_FAULT.f			
	work_path	./integ			
	bist_integ_path	./work/RP_default_spec.integ			
	group_order	BISTGRPO			
	top_module_name	top			
	TAP_hierarchy	ltop			
	nvm_ctr	2			
	connection_type	broadcast		_	
2	bist_order	RP_default			
	Log				
	Status				
	L				

Figure 5-7 Group Sub Function Block



Argument	Option		
Descri	ption		
bist_order	User defined		
This option is for users to establish the connection order of controllers in a chain.			
For example, set <b>bist_order</b> to "bist1_controller, bist2_controller, bist3_controller", and separate each MBIST controller with a comma ",". In this case, the integrating order is bist1_controller $\rightarrow$ bist2_controller $\rightarrow$ bist3_controller.			

## 5.2. Testbench Function Block

The testbench block defines testbench conditions like testbench file format, pll stable cycles and reset cycles.

The following syntax defines the testbench sub function block:

```
define{Testbench}[integration_filename]
   set pll_wait_cycle = ...
   set reset_cycle = ...
   set file_format = ...
   ...sub function block...
end_define{Testbench}
```



Option				
tegrator Test	bench BII Content			
bencn_name		_		
pll_wait_cycle 10				
reset_cycle 10				
file_format				
name	historedat			
width		_		
accert value	1161	_		
inital value	1'b1	_		
enable cycle	1			
cycle time	10	_		
cycle_time  10				
Replace	Delete Prev Next			
define{initial_sequest set width set assert_values set initial_yalues	Jence}[bistmode1] = 1			
set enable_cyc set cycle_time end_define{initial_ define{initial_sequ set width set assert_valu set enable_cyc set cycle_time end_define{initial_	ue = 1/b1 e = 1/b1 cle = 1 s=quence} [bistmode2] = 1 ue = 1/b1 de = 1/b1 de = 1 : = 10 			
set enable_cyc set cycle_time end_define{initial_ define{initial_sequ set width set assert_valu set initial_valu set initial_valu set cycle_time end_define{initial_	<pre>ue = 1'b1 de = 1'b1 e = 10 _sequence}[bistmode2] = 1 ue = 1'b1 e = 1'b1 de = 1 _sequence}</pre>			

Figure 5-8 Testbench Function Block

Argument	Option			
Description				
bench_name	User defined			
Set the test bench file name, and the default name is "INTEG_tb".				
pll_wait_cycle User defined				
Specify the stable cycle time of PLL. The MBIST circuit will be reset after these stable cycles. Default Value: 100000				
reset_cycle	User defined			
This option defines the waiting cycles to reset the MBIST circuit. While PLL is stable, the MBIST circuit will be reset after the period of <b>reset_cycle</b> .				
file_format STIL format, WGL format, Verilog				
Define the output format of testbench. The default Setting is "Verilog".				



#### 5.2.1. Initial\_sequence Sub Function Block

The Initial\_sequence sub function defines the signals on the top level which can force the system to enter testing mode. In a real chip, users may use some signals to switch function or testing mode. To run MBIST mode simulation, EZ-BIST will switch these signals to testing mode. The following syntax defines the testbench sub function block:

```
define{initial_sequence}[signal]
   set width = ...
   set assert_value = ...
   set initial_value = ...
   set enable_cycle = ...
   set cycle_time = ...
end_define{initial_sequence}
```

Figure 5-9 is the example of Initial sequence from the GUI view.

Ontion	INUS		- 0
option ntegrator Testb	ench   BII Content		
bench name	INTEG th		
pll wait cycle	10		
reset_cycle	10		
file_format	verilog		
Inital_sequence			
name	bistmode1		
width	1		
assert_value	1'b1		
inital_value	1'b1		
enable_cycle	1		
cycle_time	10		
Replace	Delete	Prev	Next
define{initial_seque set width set assert_value set initial_value set cycle_time end_define{initial_seque set width set assert_value	<pre>section []</pre>		

Figure 5-9 Initial\_sequence Sub Function Block



Argument	Option	
Description		
width	User defined	
Define the width of a signal. On the top level, users will use pins to switch function mode and testing mode.		
assert_value	User defined	
Define the assert_value while entering the testing operation.		
initial_value	User defined	
Define the initial value of the switch signal.		
enable_cycle	User defined	
The defined signal will be changed from the initial value to the asserted value after cycle values are defined with this option.		
cycle_time	User defined	
The defined signal will keep the asserted value with the cycle number which is defined in this option.		

Figure 5-10 is the example of the BII setting content from the GUI view.

e Ontion			
ntegrator   Testbench	BII Content		-
			_
define{Integrator}[INTEG]			
set port_allas			
set integrator interface	= yes		
set group order	- RISTORDO		
set ton module name	= ton		
set TAP hierarchy	= top		
set verilog path	= ./mbist/top default INS FAULT.f		
set work path	= ./integ		
set bist_integ_path	= ./mbist/top_default_spec.integ		
set work_path	= ./integ		
set nvm_ctr	=		
define (heelgup) [TD1]			
cet dedicate port	- itdi		
set mapping port	- top u pm otdi		
end define / hookun }			
end_denne{nookdp}			
define{hookup}[TMS]			
set dedicate_port	= itms		
set mapping_port	= top u_pm otms		
end_define{hookup}			
define/hookun\[TDO]			
set dedicate nort	= itdo		
set mapping port	= top u pm otdo		
end_define{hookup}			
define (heelum) (heel def	with MCK1		
cot dodicate port	aut_MCK]		
set manning port	= top MCK		
end_define{hookup}	- cop nex		
defers (headaus) (TCV)			
define{nookup}[TCK]	= itck		
set manning port	= top u pm otck		
end define (bookun)	= top u_pin otck		
end_denne(nookdp)			
define{hookup}[TRST]			
set dedicate_port	= itrst		
set mapping_port	= top u_pm otrst		
end_define{hookup}			

Figure 5-10 Example of BII Setting Content



Select and click "Run" from the "File" drop-down list to execute the BII flow as Figure 5-11 shows.

X BII Configuration Tool@VEN	- ZL	-		×
File Option				
Load BII Testber	ch   BII Content			
Save BII Alias Ho	okup			
Run				-1
evit or name	INTEG			
vernog_path	./mbist/top_default_INS_FAULT.f			
work_path	./integ			
bist_integ_path	./mbist/top_default_spec.integ			
group_order	BISTGRPO		_	
top_module_name	top		_	
TAP_hierarchy	top		_	
nvm_ctr	<b>_</b>			
Group				
connection_type	broadcast	_		
bist_order	top_default			
Log			_	
Status				
				_

Figure 5-11 Run Bll Setting File

When the BII flow is completed, the status window will pop up to inform you the result after BFL executed as Figure 5-12 shows.

integrator name	INTEG
verilog_path	./work/RP_default_INS_FAULT.f
work_path	./integ
bist_integ_path	./work/RP_default_spec.integ
group_order	BISTGRPO
top_module_name	top
nvm_ctr Group connection_type bist_order Log	RP_default RP_default RP_default BI Stage Done. Di Stage Done. PACTOR STATEST [17:16:52] [RPLACE_INST_PORT] top.u_pm.otrst (otrst, ) => (otrst, INTEG_TMS) [17:16:52] [RPLACE_INST_PORT] top.u_pm.otdo (otdo, ) => (otdo, INTEG_TOO) [17:16:52] [INSERT] Create inserted design [/home/ken.hsieh/workspace/workcase/new_NDA/integ/top _INSINS.] [17:16:52] [INSERT] Create run file for inserted BIST simulation (/home/ken.hsieh/workspace/workcase/new_NDA/integ/INT EG_INS.)
Status	(/home/ken.hsieh/workspace/workcase/new_NDA/integ/top _INS.tcl) [17:16:52] [START] Total execution time : 0.27 sec 

Figure 5-12 The Status Window When Bll Flow is Completed



# 6. Appendixes

#### 6.1. "Include" Case

For those designs, which contain a relative path with "include" and will be modified, EZ-BIST will rewrite the relative path to absolute path. Therefore, if user plan to copy the design to another path, please manually edit the absolute path based on new path or re-execute EZ-BIST to generate the correct path.

#### 6.2. Parsing Mode

If the design is RTL, please make sure it could be synthesized. Otherwise, EZ-BIST cannot parse the design for inserting MBIST circuit to the design.

Due to the diverse syntax of RTL, we suggest users using netlist as an input if RTL keeps having parsing issue.

#### 6.3. \*.rcf File

To avoid simulation failure, please use the absolute path in <code>rom.v</code> if you try to open a <code>\*.rcf</code> file.



# 6.4. Supported Testing Algorithm

Table 6-1	Testina	Algorithms	for	SRAM in	ו EZ-BIST
	rooting	/		<b>O</b> 1 (7 (11) 11	

Memory Type	Name	Fault Detection	Algorithm
SRAM	March CW (part 1)	SAF, TF, AF, CFin, CFid, CFst, SOF, RDF	>(wa) >(ra,wb) >(rb,wa,ra) <(ra,wb,rb) <(rb,wa) <(ra)
	March CW (part 2)	Word-oriented CF	>(wa) >(wb) >(rb,wa,ra)
	March Y	SAF, TF, CFin, SOF, RDF	>(wa) >(ra,wb,rb) <(rb,wa,ra) <(ra)
	March X	SAF, TF, AF, CFin	>(wa) >(ra,wb) <(rb,wa) <(ra)
	MATS++	SAF, TF, AF, SOF	>(wa) >(ra,wb) <(rb,wa,ra)
	MOVI	SAF, TF, AF, CFin, CFst, SOF, RDF	<(wa) >(ra,wb,rb) >(rb,wa,ra) <(ra,wb,rb) <(rb,wa,ra)
	Ext March C-	SAF, TF, AF, CFin, CFid, CFst, SOF	>(wa) >(ra,wb) >(rb,wa,ra) <(ra,wb) <(rb,wa) <(ra)
	*March C+	SAF, TF, AF, CFin, CFid, CFst, SOF, RDF	>(wa) >(ra,wb,rb) >(rb,wa,ra) <(ra,wb,rb) <(rb,wa,ra) <(ra)
	March C-	SAF, TF, AF, CFin, CFid, CFst	>(wa) >(ra,wb) >(rb,wa) <(ra,wb) <(rb,wa) <(ra)
	March C Gray	ADOF	>(wa) >(ra,wb) >(rb,wa) <(ra,wb) <(rb,wa) <(ra) Address only one bit change
	March LR	SAF, TF, AF, CFin, CFid, CFst, SOF	>(wa) >(ra,wb) >(rb,wa,ra,wb) >(rb,wa) >(ra,wb,rb,wa) >(ra)
	March C	SAF, TF, AF, CFin, CFid, CFst	>(wa) >(ra,wb) >(rb,wa) >(ra) <(ra,wb) <(rb,wa) <(ra)
	March B	SAF, TF, AF, CFin, CFid, SOF	>(wa) >(ra,wb,rb,wa,ra,wb) >(rb,wa,wb) <(rb,wa,wb,wa) <(ra,wb,wa)
	March A	SAF, TF, AF, CFin, CFid	>(wa) >(ra,wb,wa,wb) >(rb,wa,wb) <(rb,wa,wb,wa) <(ra,wb,wa)
	March 17N	SAF, TF, AF, CFin, CFid, CFst, SOF, RDF	>(wb) >(rb,wa,ra) >(ra,wb,rb) >(rb,wa) <(ra,wb,rb) >(rb) <(rb,wa,ra) >(ra)



March 19N	'SAF', 'TF', 'AF', 'CFin', 'CFid', 'CFst', 'SOF', 'RDF'	>(wa,ra) >(wa) >(ra,wb,rb) >(rb) >(rb,wa,ra) >(ra) <(ra,wb,rb) >(rb) <(rb,wa,ra) >(ra)
March 33N	dRDF, dIRF, dDRDF, dTF, dWDF	>(wa) >(wa,wb,wa,wb) >(rb,wa,wa) >(wa,wa) >(ra,wb,rb,wb,rb,rb) <(rb) <(wb, wa,wb,wa) <(ra,wb,wb) <(wb,wb) <(rb,wa,ra,wa,ra,ra) <(ra)
March 33N-	'dRDF', 'dIRF','dDRDF', 'dTF', 'dWDF'	'>(wa) >(wa,wb,wa,wb) >(r- 1b,wa,wa) >(wa,wa) >(r-1a,wb,r- 1b,wb,r-1b,r-1b) <(r-1b) <(wb,wa,wb,wa) <(r-1a,wb,wb) <(wb,wb) <(r-1b,wa,r-1a,wa,r- 1a,r-1a) <(r-1a)'
March M	SAF, TF, AF, CFin, CFid, CFst, SOF, RDF	>(wa) >(ra,wb,rb,wa) >(ra) >(ra,wb) >(rb) >(rb,wa,ra,wb) >(rb) <(rb,wa)
March Mdsn1	SAF, TF, AF, CFin, CFid, CFst RET	Part1~Part4
March Mdsn1 (part1)	SAF, TF, AF, CFin, CFid, CFst	>(wa) >(wb,wa) (SLP) >(ra,wb,wb)
March Mdsn1 (part2)	SAF, TF, AF, CFin, CFid, CFst	>(rb,wa,ra,wa,ra,wb) >(rb,rb)
March Mdsn1 (part3)	SAF, TF, AF, CFin, CFid, CFst	<(wa,wb) (SLP) <(rb,wa,wa)
March Mdsn1 (part4)	SAF, TF, AF, CFin, CFid, CFst	<(ra,wb,rb,wb,rb,wa) <(ra,ra)
March SSSc	SAF, TF, AF, CFin, CFid, CFst	>(wa) >(wb,wb,rb,rb,wa) >(wb) >(wb,wb,rb,rb,wa)
Non-March BM	detect bit/group write enable faults and datapath shorts.	>(wa) >(wB5b,rB5b) <(wBAb,rBFb) >(wBAa,rBAa) <(wB5a,rBFa)
MARCH_RET	RET	<(wb) (SLP) <(rb) >(wa) (SLP) >(ra)
СВ	BF	>(wa) >(ra) >(wb) >(rb)
March 8R	dRDF	>(wa,ra,ra,ra,ra,ra,ra,ra,ra) >(wb,rb,rb,rb,rb,rb,rb,rb,rb)
March 5W	SAF, TF, CFst, dWDF, WDF	>(wa) >(ra,wb,rb,wb,wb,wb,wb)

Copyright© iSTART-TEK INC Limited 2023-2024, All Rights Reserved.

November, 2023



		>(rb,wa,wa,wa,wa,wa) <(ra,wb,wb,wb,wb,wb) <(rb,wa,wa,wa,wa,wa)
March RP	WDF	>(wa) >(ra,wb) >(rb,wa,r-1a) <(ra,wb,r-1b) <(rb,wa) >(ra)
**March d2PF	'SAF', 'TF', 'AF', 'CFin', 'CFid', 'CFst', 'SOF', 'RDF', 'Weak WL', '2PFavS'	'>(n wa) >(r+1a n,n wb) >(r+1b n,n wb) >(r+1b n,n wa) >(r+1a n,n wa) >(r+1a n,n wb) >(r+1b n,n wb) >(r+1b n,n wa) >(r+1a n,n wa)'
**March s2PF	'SAF', 'TF', 'AF', 'CFin', 'CFid', 'CFst', 'SOF', 'RDF', 'Weak WL', '2PF1s', '2PF1as'	'>(n wa) >(ra n,ra n, n wb) >(rb n, rb n, n wa) <(ra n, ra n, n wb) <(rb n, rb n, n wa) <(ra n)'
***March A2PF-M	SAF, TF, AF, CFin, CFid, CFst, SOF, RDF, Weak WL, A2PF	>(wa n) >(ra ra,wb r+1a,wb r- 1b,rb rb) >(rb rb,wa r+1b,wa r- 1a,ra ra) <(ra ra,wb r- 1a,wb r+1b,rb rb) <(rb rb,wa r- 1b,wa r+1a,ra ra) <(ra n)

\*: Default testing algorithm of EZ-BIST

\*\* : Support two port memory only

\*\*\* : Support dual port memory only

±1: used to increase/ decrease memory address

|: used to separate operation of different port

>: indicates address count from o to the highest address in a memory.

<: indicates address count from the highest address to 0 in a memory.

a: indicates test pattern.

b: indicates inverse "a" test pattern.

MISR (Multiple-Input Signature Register)

LFSR (linear feedback shift register)

Table 6-2 Testing Algorithms for ROM in EZ-BIST

Memory Type	Name	Address sequence	Operation	Description
DOM		LFSR	(rc)	Reads and compresses ROM's content
ROM	KOW Test	N/A	Compare MISR	Compares the final signature



#### 6.5. Statistics in TSMC SP Memory

#### **Design Architecture**:

- ✓ Memory: Single-port SRAM \*20 and ROM \*1
- ✓ **Process**: TSMC 55nm
- ✓ Library: sc9\_cln55lp\_base\_rvt\_ss\_typical\_max\_1p08v\_125c
- ✓ **NAND Gate area**: 1.44 um<sup>2</sup>

#### I. The default setting of BFL file: default.bfl

BFL File Column	Default Value
clock_trace	no
STIL_test_bench	no
asynchronous_reset	yes
bist_interface	basic
address_fast_y	no
algorithm_selection	no
background_style	SOLID
background_bit_inverse	no
background_col_inverse	no
bypass_support	no
bypass_clock	no
bypass_reg_sharing	1
clock_function_hookup	no
clock_switch_of_memory	yes
clock_source_switch	no
clock_within_pll	no
diagnosis_support	no
diagnosis_data_sharing	no
diagnosis_memory_info	no
diagnosis_time_info	no
diagnosis_faulty_items	all
parallel_on	no
reduce_address_simulation	no
rom_result_shiftout	no
Q_pipeline	no
algorithm	March C+

#### Table 6-3 The Default Setting of BFL file



BFL File Column	Default Value
meminfo	1 Ctr 2 Seq 2 Group

#### Table 6-4 Synthetic Area of default.bfl

Referenced Library	Total Area
top_default_controller	798.120025
top_default_sequencer1	528.84001
top_default_sequencer2	258.480007
top_default_ter_1_1_1	402.840007
top_default_ter_1_1_2	402.840007
top_default_ter_1_1_3	402.840007
top_default_ter_1_1_4	402.840007
top_default_ter_1_1_5	402.840007
top_default_ter_1_1_6	402.840007
top_default_ter_1_1_7	402.840007
top_default_ter_1_1_8	402.840007
top_default_ter_1_1_9	402.840007
top_default_ter_1_1_10	402.840007
top_default_ter_1_1_1	402.840007
top_default_ter_1_1_12	402.840007
top_default_ter_1_1_13	402.840007
top_default_ter_1_1_14	402.840007
top_default_ter_1_1_15	402.840007
top_default_ter_1_1_16	402.840007
top_default_ter_1_1_17	402.840007
top_default_ter_1_1_18	402.840007
top_default_ter_1_1_19	402.840007
top_default_ter_1_1_20	402.840007
top_default_ter_2_1_1	164.160006
top_default_tpg_1_1_1	334.8
top_default_tpg_1_1_2	336.24
top_default_tpg_1_1_3	336.24
top_default_tpg_1_1_4	334.8
top_default_tpg_1_1_5	333.36
top_default_tpg_1_1_6	334.8
top_default_tpg_1_1_7	333.36



Referenced Library	Total Area
top_default_tpg_1_1_8	334.8
top_default_tpg_1_1_9	331.92
top_default_tpg_1_1_10	333.36
top_default_tpg_1_1_1	334.8
top_default_tpg_1_1_12	334.8
top_default_tpg_1_1_13	334.8
top_default_tpg_1_1_14	334.8
top_default_tpg_1_1_15	333.36
top_default_tpg_1_1_16	333.36
top_default_tpg_1_1_17	333.36
top_default_tpg_1_1_18	331.92
top_default_tpg_1_1_19	331.92
top_default_tpg_1_1_20	331.92
top_default_tpg_2_1_1	606.960003

Total 145 references	17092.08019

(Unit:um2)



# II. Refer to Circuit Area Comparison table to change each option in default.bfl file.

For example, set the option **asynchronous\_reset** to "no", the circuit area will become 99.085% of the original circuit area, which means the circuit area will decrease by about 0.91%.

Process/Lib.: TSMC 55nm/ sc9_cln55lp_base_rvt_ss_typical_max_1p08v_125c +: Increase, -: Decrease	Default .bfl
asynchronous_reset = no	-0.91%
address_fast_y = yes	3.08%
clock_within_pll = yes	0.11%
parallel_on =yes	0.96%
reduce_address_simulation = yes	3.10%
rom_result_shiftout = yes	7.05%
Q_pipeline = yes	67.60%

#### Table 6-5 Area Comparison Table

bist_interface = ieee1500	1.03%
bist_interface = ieee1149.1	2.25%

algorithm add March C-	0.44%
algorithm add March C- algorithm_selection = outside	0.61%
algorithm add March C- algorithm_selection = scan	0.61%

Note: If the option **algorithm\_selection** set to "outside" or "scan", the circuit area will increase by 0.17%.

Process/Lib.: TSMC 55nm/ sc9_cln55lp_base_rvt_ss_typical_max_1p08v_125c +: Increase, -: Decrease	Default .bfl
background_style = 5A	0.93%
background_bit_inverse = yes	2.07%
background_col_inverse = yes	0.67%

bypass_support = wire	14.42%
bypass_support = reg	82.81%
bypass_support = reg	82 81%
bypass_clock = yes	02.0170
bypass_support = reg	
bypass_clock = yes	60.38%
bypass_reg_sharing = 2	
bypass_support = reg	
bypass_clock = yes	45.74%
bypass_reg_sharing = 4	

Note: If the option **bypass\_support** is set to "reg", the circuit area will increase by 82.81%. If the option **bypass\_clock** is set to "yes", the circuit area will increase by 82.81%. However, if **bypass\_reg\_sharing** is set to "2", the circuit area will only increase by 60.38%. The option **bypass\_reg\_sharing** can effectively reduce the circuit area.



### 6.6. RTL Syntax Restrictions

I. For a module instance, empty port information is not allowed. Example:

> module UART (D, Q, CK); input D, CK; ... endmodule

The following syntax is not supported:

UART u\_uart();

Instead, the following syntax is supported:

UART u\_uart(.CK());

II. A module with no content inside is not supported. A module must have at least one line of RTL code inside. Example:

*module wrapper (input ck); endmodule*